MASTER EN SCIENCES INFORMATIQUES

Production de la documentation d'une base de données sous la forme d'un hypertexte

Furnelle, Jacques

Award date:
2004

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique.
Année académique 2003-2004

Production de la documentation d'une base de données sous la forme d'un hypertexte

Jacques Furnelle

Mémoire présenté en vue de l'obtention du grade de Licencié en informatique

Résumé

Le développement des bases de données est une activité complexe, qui requiert la collaboration de spécialistes à chaque étape du processus. Dans ce contexte, la production et le maintien d'une documentation de qualité sont essentiels pour assurer une bonne communication entre les intervenants. Nous proposons, dans ce mémoire, d'aborder le besoin de documentation pour cette activité, et en particulier d'étudier le besoin de documentation pour les schémas qui modélisent ces systèmes. Sur base des éléments de cette étude, un générateur de documentation de type hypertexte est proposé pour l'atelier DB-Main. Ce générateur utilise le langage de balisage extensible XML et les techniques de transformation qui lui sont associées, pour produire automatiquement la documentation. Au travers de cette démarche, nous tentons de mettre en évidence l'intérêt des formats de publication de type hypertexte ainsi que les avantages de XML pour la documentation des bases de données.

Mots clés : *Documentation, documentation technique, bases de données, hypertexte, XML, maintenance logicielle, acquisition des connaissances, outil CASE.*

Abstract

Database development is a complex activity, which requires specialists collaboration at each stage of the process. Within this context, production and maintenance of quality documentation are important to ensure a good communication between intervening parties. In this document, we propose to study documentation requirements for this activity and in particular documentation relate to schemas which provide model of database. Basis on this study, we also present hypertext documentation generator dedicate to DB-Main case tool. This generator, take advantage of extensible markup language XML and transformation techniques related in order to automatically produce documentation. By this, we intend to highlight benefits of hypertext publication format and XML for database documentation activity.

Keywords: *Documentation, technical documentation, database,* hypertext, *XML, software maintenance, knowledge acquisition, tool CASE.*

Remerciements

Je tiens d'abord à remercier l'équipe du Laboratoire d'Ingénierie des Bases de Données dans son ensemble pour son accueil chaleureux.

Je souhaite bien sûr, remercier plus particulièrement le Professeur Jean-Luc Hainaut, mon promoteur, pour la confiance qu'il m'a accordé en me confiant ce sujet de mémoire et pour ses excellents conseils.

Dans le même esprit, je tiens également à manifester ma reconnaissance à Jean Henrard, de l'équipe du LIBD, qui m'a suivi lors de la réalisation du mémoire. Son aide, sa disponibilité ainsi que ses nombreux conseils m'ont beaucoup aidé tout au long de ce travail.

Je remercie enfin tous ceux qui, de près ou de loin, ont contribué au bon déroulement de ce mémoire et bien sûr ma famille qui m'a toujours soutenu.

Table des matières

Table des figures

1. Introduction

Avec l'introduction massive de systèmes d'information dans les organisations, le parc des bases de données actuellement en activité s'est considérablement développé et avec cette croissance, sont apparues des difficultés pour maîtriser et faire évoluer ces systèmes. Outre la complexité intrinsèque des systèmes informatiques, les difficultés rencontrées pour assurer la maintenance des systèmes de base de données sont en grande partie liées à la mauvaise qualité des documentations disponibles.

L'objectif du mémoire présenté ici, est d'étudier le besoin des développeurs en ce qui concerne la documentation des grands schémas de bases de données et, sur base de cette étude de proposer un générateur de documentation de type hypertexte pour l'atelier DB-Main. L'étude consiste, en fait, à comprendre le besoin de ceux qui produisent et consultent la documentation relative aux bases de données et plus particulièrement aux schémas qui modélisent ces systèmes. Pour cette étude, nous devons tenir compte de la contrainte introduite par les tailles importantes des systèmes actuels, en effet, il n'est pas rare que les bases de données des entreprises comportent plusieurs milliers de tables réparties sur différents systèmes [Hainaut 00]. L'outil proposé pour l'atelier DB-Main devra quant à lui, répondre aux besoins de professionnels qui sont d'une part les concepteurs de bases de données et d'autre part les programmeurs qui utilisent ces bases de données pour construire leurs applications.

Pour prendre la mesure du problème posé, la question de la documentation, doit être replacée dans le contexte de la maintenance logicielle. En effet, les activités de maintenance et d'intégration des applications constituent aujourd'hui un enjeu capital pour les entreprises. Avec des coûts pouvant représenter plus de 70 % des montants investis [Wolak 01] et avec plus de la moitié des développeurs affectés au maintien et à la modification des applications existantes, cette activité tend à occuper une place prépondérante dans le secteur informatique. Il ressort de ce constat, que la maîtrise du processus de la maintenance logicielle est devenue fondamentale pour les équipes en charge des systèmes informatiques.

Pour répondre à ce défi, parmi les moyens disponibles pour soutenir ce processus, la documentation des applications demeure indéniablement un élément fondamental. La disponibilité d'une documentation de qualité constitue, nous le verrons, un avantage déterminant qui se traduit par des gains de productivité appréciables lors de l'activité. Son absence est par contre, une des premières causes d'échec des projets qui ont pour objectif de faire évoluer les systèmes.

Pourtant, malgré l'importance du besoin, les études réalisées dans ce domaine mettent en évidence que la production et le maintien de la documentation des applications constituent, aujourd'hui encore, un point faible du développement informatique.

Pour l'activité de développement liée aux bases de données, le constat reste le même, mais il prend ici davantage encore d'importance. En effet, si l'existence d'un SGBD est de l'ordre de 4 ans [Hainaut 00], la durée de vie de la base de données exploitée est elle beaucoup plus longue. Pour les entreprises, les données gérées par ces systèmes constituent bien souvent un historique de leurs activités qui sera exploité durant plusieurs dizaines d'années. L'information gérée par ces systèmes est donc essentielle pour le bon fonctionnement de l'entreprise et le souhait de maintenir ces bases de données actives suffit à lui seul pour justifier la mise en place d'un système de documentation fiable pour celles-ci.

Bien sûr, d'autres motivations conduisent également à souhaiter produire et maintenir une telle documentation. Nous aurons évidemment le désir de limiter les coûts liés à la maintenance, mais aussi, le souhait de conserver une connaissance précise des systèmes pour permettre leurs évolutions et leurs adaptations aux nouveaux besoins de l'entreprise. Des opérations lourdes telles que l'intégration de bases de données anciennes ou hétérogènes ou encore les migrations vers de nouvelles architectures, nécessitent de bien connaître l'infrastructure en place et pour les bases de données, de disposer de schémas bien documentés. Le besoin de documenter ces systèmes est d'ailleurs particulièrement bien mis en évidence par l'importance croissante de l'activité de rétro-ingénierie dont l'un des objectifs premiers est la reconstitution d'une documentation pertinente.

Face à cette question de la documentation, une démarche volontariste constitue certainement un point de départ intéressant, mais le seul souhait de produire des documents ne suffit pas à résoudre le problème. Pour fournir une documentation vraiment utile, nous devons nous intéresser aux besoins rencontrés par les lecteurs qui sont appelés à consulter cette documentation. Une démarche contraire risquerait d'entraîner le rejet de la documentation produite et donc l'abandon de l'outil documentaire proposé. Ceci est évidemment le cas pour la documentation des bases de données, en effet, les volumes de documents à produire, et donc à consulter, sont ici particulièrement importants. Une manière de rencontrer cette contrainte, est de proposer une documentation de type hypertexte qui devrait permettre au lecteur de consulter rapidement et facilement les documents via le navigateur installé sur son poste de travail. C'est cette approche que nous nous proposons d'explorer durant cette étude.

Pour mener à bien notre travail, nous tentons dans un premier temps de définir les qualités d'une bonne documentation, et pour ce faire, nous nous appuyons sur plusieurs enquêtes publiées dans ce domaine. Nous passons ensuite en revue les solutions existantes et sur base des éléments relevés, nous proposons une solution pour le générateur de documentation de DB-Main. L'implémentation d'un prototype de la solution est enfin présentée.

Notre document se présente comme suit :

- La première partie présente l'étude du besoin de documentation. Après avoir exposé la problématique de la documentation et avoir défini ce besoin pour l'activité de développement des bases de données, nous tentons, sur base des publications consultées, de proposer un ensemble de qualités auxquelles devrait répondre la solution proposée.

- Dans la seconde partie du document, nous présentons un état de l'art des technologies de l'informatique documentaire. Au cours de cette présentation, nous mettons particulièrement l'accent sur les avantages et les implications de la mise en œuvre des formats de documents structurés. L'utilisation de ces formats semble en effet, constituer un point commun des technologies actuellement adoptées.

- La troisième partie présente la solution retenue pour l'outil de documentation de DB-Main. Après avoir effectué l'analyse des besoins de l'outil ainsi qu'avoir étudié les éléments existants, la solution est présentée avec une justification des choix effectués. Nous détaillons dans cette partie du document, les différentes stratégies retenues, notamment pour la documentation de la traçabilité Forward/Backward et nous abordons également le principe de séparation du contenu, de la présentation et de la publication qui guidera notre travail durant le développement.

- La quatrième et dernière partie expose l'architecture ainsi que les détails d'implémentation et de fonctionnement du prototype. Nous terminons cette quatrième partie par une présentation des résultats obtenus lors de l'utilisation de l'outil de documentation avec des exemples de projets DB-Main.

- Nous concluons ce mémoire en dressant le bilan de l'ensemble de la démarche et en présentant les faits saillants mis en évidence durant ce travail. Nous proposons également des extensions possibles de ce travail.

2. *Etude du besoin de documentation*

Dans ce chapitre, nous introduisons tout d'abord la problématique de la documentation dans le contexte de la maintenance logicielle, nous abordons ensuite les besoins de documentation rencontrés par les développeurs dans le cadre de l'activité de développement des bases de données. Nous tentons enfin de dégager les qualités que devrait présenter la documentation produite pour cette activité.

2.1. Le besoin de documentation dans la maintenance logicielle

2.1.1. L'enjeu de la maintenance logicielle

Dans le cycle de vie des SI, la maintenance constitue à l'évidence la phase la plus longue du processus logiciel. Il en résulte qu'aujourd'hui, la maintenance et l'amélioration des systèmes existants mobilisent 65 à 75 % des budgets et représentent la part la plus importante de l'activité informatique [Wolak 01]. Outre l'importance des budgets engagés, la maîtrise de la maintenance des systèmes d'information peut également impacter de manière considérable l'activité des entreprises, et ce, pour les raisons suivantes :

- Dans un environnement économique changeant, la rapidité d'adaptation des entreprises est devenue primordiale. Cette capacité d'adaptation est évidemment conditionnée par leur faculté à faire évoluer leurs infrastructures informatiques. Il s'agit ici de maîtriser la maintenance évolutive.

- Les souhaits d'amélioration des services ou de réduction des coûts imposent de plus en plus souvent des changements d'infrastructures et des opérations d'intégration de systèmes qui impliquent un contrôle réel de l'environnement et de la maintenance adaptative.

- Avec la mise en place des réseaux et le mouvement d'ouverture des entreprises vers leurs partenaires et leurs clients, celles-ci exposent leur savoir-faire en développant des services en lignes. Dans ce contexte, la maîtrise du processus logiciel et des activités de maintenance est essentielle, notamment en ce qui concerne la maintenance corrective.

L'enjeu de la maintenance est donc devenu capital pour les entreprises, en termes de coût bien sûr, mais également en termes de capacité d'adaptation à l'environnement économique.

2.1.2. Le besoin de documentation

Comprendre les applications et les systèmes informatiques conçus par d'autres constitue le point de départ de l'activité de maintenance. Le transfert de connaissances entre les équipes, est donc un point de passage obligé pour cette activité.

Dans ce contexte, l'existence d'une documentation pertinente apparaît comme un facteur essentiel pour le processus de maintenance. Pour Glass, il s'agit même d'un des principaux facteurs d'amélioration du processus [Glass 89]. Pour d'autres, la documentation est plutôt vue comme un moyen d'amélioration de la productivité de l'activité, qui permet par exemple, des gains de temps significatifs [Tryggeseth 97].

Pour ces auteurs, comme pour beaucoup d'autres, la disponibilité d'une documentation de qualité constitue bien un besoin évident pour l'activité de maintenance.

2.1.3. La problématique de la documentation dans la maintenance logicielle

Malgré la mise en évidence d'un besoin de documentation clairement établi, les enquêtes menées dans ce domaine font pourtant apparaître une situation de terrain bien peu satisfaisante.

L'étude de Chapin [Chapin 85] présente la pauvreté de la documentation disponible comme le plus important problème rencontré par les équipes en charge de la maintenance des applications. Cette mauvaise qualité pouvant résulter d'une des caractéristiques suivantes.

- Non-existence de la documentation.
- Incohérence entre les documents et les systèmes qu'ils décrivent.
- Difficultés d'accès à la documentation.
- Type de documentation non adaptée aux niveaux et aux besoins des personnes.

Ce constat se trouve confirmé par d'autres études plus récentes dont une enquête de l'IEEE qui indique que 90 % des logiciels ont une documentation incomplète ou obsolète et que dans 89 % des cas les développeurs en sont réduits à utiliser le code disponible comme principale source d'information [Sousa 98]. Il en résulte que la majorité du temps consacré à la maintenance est passé non pas à modifier le code, mais à rechercher l'information et à essayer de la comprendre.

Dans le contexte de l'activité de maintenance, cette enquête établit les 3 facteurs les plus négatifs rencontrés comme étant :

1. L'absence de processus de maintenance logicielle.
2. Le manque de documentation.
3. Le manque de temps pour mettre à jour la documentation.

2.2. Le besoin de documentation pour l'activité de développement des bases de données

2.2.1. L'activité de développement des bases de données

Le développement des bases de données et des applications qui les utilisent, suivent le cycle de développement habituel des systèmes d'information. Nous retrouvons ici aussi, la succession des étapes de développement que sont l'analyse des besoins, l'analyse, la spécification, le design, l'implémentation, le déploiement et la maintenance.

Pour la conception de bases de données, ce processus peut en outre s'appuyer sur des méthodologies de construction qui sont propres au domaine et parmi les techniques existantes, la conception par transformations de schéma constitue certainement une méthode de développement éprouvée pour la construction et la validation des bases de données [Hainaut 00].

Dans cette approche, le cycle de développement peut être redéfini de la manière suivante :

Nous présentons ici, de manière simplifiée, les étapes de la méthode.

- L'étude des besoins, qui comprend la prise de connaissance du domaine d'application de la base de données et la définition de l'objectif du système.
- L'analyse conceptuelle qui est l'étape de modélisation du domaine. Cette étape débouche sur l'élaboration d'un schéma conceptuel.
- La conception logique qui permet de dériver le schéma conceptuel vers un schéma logique compatible avec le modèle logique du système, le modèle relationnel dans notre cas.
- La conception physique et le codage, le schéma logique est ici adapté pour intégrer les spécificités du SGBD mis en œuvre. Nous obtenons avec cette étape, un schéma physique qui spécifie complètement la base de données à implémenter et qui permettra la génération du code.
- L'implantation des données dans le système.
- Le développement des applications qui utilisent la base de données.
- La mise en production.
- La maintenance et évolution du système.

Le bref exposé ci-dessus nous montre qu'il s'agit en fait, d'un ensemble de tâches très spécifiques et qui requièrent des compétences techniques importantes pour être menées à bien. C'est donc à des spécialistes de ces technologies que seront confiées les différentes étapes du développement.

A côté des activités visant à la mise en place de nouveaux systèmes, nous assistons également à l'émergence d'activités nouvelles telles que la réingénierie et la rétro-ingénierie des bases de données. Ces activités, dont l'objectif est l'étude et l'amélioration des systèmes existants, connaissent actuellement une forte croissance liée à l'importance du parc de systèmes de bases de données existant et dont on souhaite l'évolution.

Notons également que pour mener à bien ces différentes activités, les développeurs sont aidés dans leur travail par des d'outils CASE qui leurs apportent une assistance, en particulier pour les étapes de modélisation et de validation. Ces outils permettent aussi d'automatiser certaines tâches comme la génération du code. L'atelier DB-Main est un de ces outils CASE.

2.2.2. Le besoin de documentation

La mise en place des activités exposées ci-dessus, suppose bien évidemment, l'organisation d'un travail en équipe ce qui implique des transferts d'informations entre spécialistes. Ceci se traduit naturellement par un besoin de produire et de consulter des documents de référence.

Il est intéressant de remarquer que les besoins de documentation rencontrés par les programmeurs qui utilisent les bases de données pour développer leurs applications, s'apparentent assez aux besoins de documentation rencontrés par les équipes en charge de la maintenance évolutive. Il s'agit en effet, dans les 2 cas, d'exploiter des systèmes existants pour développer de nouvelles fonctionnalités.

On peut également souligner que le développement actuel de l'activité de rétro-ingénierie des systèmes de base de données est sans doute lié à la mauvaise qualité ou à l'absence de documentation de ces systèmes [Jahnke 99].

En fait, comme pour l'activité de maintenance logicielle, l'absence de documentation impacte de manière très négative la maintenance des bases de données [Hainaut 00].

2.2.3. Les besoins de documentation spécifiques et les contraintes

Générer des documents utiles pour ces activités, implique bien entendu de prendre en compte les besoins spécifiques de l'ingénierie des bases de données.

2.2.3.1. Le besoin de documenter les schémas

Le premier de ces besoins, est de disposer d'une documentation précise des schémas qui décrivent la base donnée. La documentation relative aux schémas constitue un prérequis à l'objectif de développement et de maintenance des bases de données [Jahnke 99]. En particulier, la documentation des schémas conceptuels est essentielle pour comprendre le système étudié, car ce sont ces schémas qui modélisent le domaine et les règles de fonctionnement du système étudié.

Ce besoin de documenter les schémas est également bien mis en évidence par le développement de l'activité de rétro-ingénierie mentionnée précédemment, car un des objectifs premiers de cette activité est la reconstruction des schémas conceptuels et des schémas logiques.

2.2.3.2. La traçabilité des opérations de transformations de schéma

Avec les techniques de transformations de schéma, nous avons l'introduction d'un besoin de documentation particulier qui est de documenter ces transformations. L'objectif poursuivi par ce type de documentation n'est pas de fournir directement de l'information sur l'application étudiée, mais de fournir des indications relatives au processus de développement utilisé. Par exemple, on souhaite pouvoir comprendre de quel objet conceptuel est issu telle table d'un schéma logique, ou encore, qu'elles sont les transformations subies par tel type d'association pour obtenir son implémentation finale.

2.2.3.3. Les contraintes de taille et de complexité des grands schémas

Avec la mise en place des grands systèmes de gestion, les bases de données des entreprises peuvent aujourd'hui atteindre des tailles considérables. Ces systèmes peuvent en effet contenir plusieurs milliers de tables qui se répartissent sur plusieurs bases de données [Hainaut 00]. Ces grands volumes s'accompagnent également d'une complexité importante, l'origine de cette complexité pouvant par exemple résulter d'opérations d'intégration successives. Une conséquence importante de cette situation est que les schémas qui décrivent ces systèmes sont également de taille très importante et que ceci se traduit par la nécessité de produire et de consulter de grands volumes de documentation.

2.3. Qualité de la documentation

Les publications consultées sur ce sujet mettent en évidence que la documentation doit être produite sur base des besoins des utilisateurs de celle-ci. En effet, un critère qui nous semble pertinent pour caractériser une bonne documentation est que cette documentation soit effectivement utilisée.

Produire des documents qui répondent aux besoins des utilisateurs implique donc de se poser les deux questions suivantes :

1. Qui sont les utilisateurs ?
2. Quels sont les souhaits de ces utilisateurs ?

2.3.1. Qui sont les utilisateurs ?

La réponse à cette première question est, pour nous, fournie par l'objectif que nous avons défini pour ce travail. Il s'agit de proposer un outil de documentation destiné aux concepteurs et aux développeurs de base de données. Par concepteurs, nous entendons les personnes qui sur base de leurs analyses et de leurs connaissances du domaine sont amenées à définir le design de la base de données, il s'agit notamment de définir les schémas conceptuels et les schémas logiques. Les développeurs vont eux, sur base du design défini, implémenter la base de données ou encore utiliser cette base de données pour développer leurs applications.

2.3.2. Quels sont les souhaits de ces utilisateurs ?

Pour répondre à cette question, nous nous appuyons sur une importante enquête menée auprès de professionnels. Cette enquête : « *Qualities of Relevant Software Documentation - An Industial Study* » d'Andrew Forward [Forward 02a] présente quelques conclusions intéressantes sur les souhaits de ces professionnels :
Pour ceux-ci, les attributs d'une documentation efficace sont les suivants :

- La qualité du contenu de la documentation : Il s'agit du facteur le plus important, le contenu de la documentation doit rencontrer le besoin du lecteur qui la consulte.
- Le niveau de mise à jour des documents : Le second souhait est de disposer d'une documentation à jour qui corresponde réellement à la situation des systèmes décrits.
- La disponibilité et la facilité d'exploitation des documents : La rapidité d'accès à l'information est aussi un souhait fréquemment mentionné. Ce point est évidemment fort important pour la documentation des grands systèmes.
- L'utilisation d'exemples : La présence d'exemples dans la documentation permet d'illustrer le fonctionnement du code et facilite sa compréhension.

Cette étude met également en évidence l'intérêt porté aux techniques d'extraction automatique comme source d'information pour la documentation [Forward 02b].
D'autres publications [Hartmann 01], [Reiter 95] confirment cet intérêt pour l'automatisation de la production de la documentation, notamment pour des raisons de coût et de rapidité de mise à jour.

Les résultats de ces études nous conduisent à proposer les qualités de documentation suivantes :

2.3.3. La qualité du contenu de la documentation

L'échec des normes traitant du contenu des documentations, confirme l'idée que l'attente des utilisateurs en la matière est de disposer d'informations parfaitement adaptées à leurs besoins et qu'une démarche généraliste n'est pas toujours souhaitable. Un autre aspect également mis en avant par l'utilisation de ces normes est qu'un volume excessif de documentation peut constituer un facteur de rejet de celle-ci, car trop de documentation nuit à la qualité de la documentation.

Nous pensons toutefois pouvoir dégager 2 types d'informations qu'il nous semble souhaitable de pouvoir collecter. En effet, les bases de données comme tous les systèmes d'information sont par définition constituées d'une composante informatisée, le système, et d'une composante humaine ou organisationnelle. L'outil de documentation proposé devra donc permettre de collecter ces deux types d'information :

2.3.3.1. La documentation du système

Il s'agit donc d'information sur le système lui-même, sa description, que l'on souhaite obtenir le plus à jour possible. Cette documentation devra fournir toutes les informations nécessaires pour permettre de comprendre le dispositif, avec notamment :

- Les composants du système.
- Les relations entre les composants et à l'intérieur des composants.
- Les catalogues et les dictionnaires de données.

Sur le plan de sa production, c'est cette partie de la documentation qui se prête le mieux à une collecte automatisée de l'information.

2.3.3.2. Les informations relatives à la composante humaine et à l'organisation

Ces informations sont souvent détenues par les acteurs qui participent au développement, nous retrouvons parmi ces personnes, les experts du domaine, les analystes et les développeurs de la base de données. Les connaissances de ces spécialistes constituent une source d'information de premier ordre et la capture de ces connaissances, un objectif de l'outil documentaire à développer. L'intégration d'interfaces d'acquisition permettant l'introduction aisée de ces informations au niveau des outils Case constitue sans doute une piste intéressante à suivre. La capacité d'intégration de documents de diverses natures est également importante, car des documents tels que : le cahier des charges, les documents de gestion des managers de projet, les rapports d'interview, des résultats de tests, etc. peuvent s'avérer particulièrement utiles. [Forward 02 a] et [Hartmann 01]

2.3.4. La disponibilité et la facilité d'exploitation

La seule existence d'une documentation pertinente et à jour n'est pas suffisante pour atteindre l'objectif fixé [Forward 02a]. La disponibilité des informations et la facilité d'exploitation jouent également un rôle prépondérant dans l'adoption d'un outil documentaire. Ceci implique la mise en place d'infrastructures ou l'adoption de méthodes permettant :

- Une gestion efficace et conviviale de la documentation produite. Il est important de savoir où et comment trouver les documents.

- Une prise en main rapide qui ne nécessite pas d'apprentissage supplémentaire. Pour cet aspect, l'utilisation des navigateurs Web comme outil de consultation semble assez indiquée pour un public constitué de professionnels de l'informatique.

- Un mode consultation adapté au type de document. Pour une documentation volumineuse, une attention toute particulière doit être apportée à la structure des documents et à la facilité de navigation.

2.3.5. L'automatisation

La production de documentation reste une activité coûteuse, car grande consommatrice de ressources et de temps, c'est d'ailleurs cette raison qui est la plus souvent invoquée pour justifier les lacunes dans ce domaine [Sousa 98]. Mais outre les considérations de coûts, l'aspect laborieux et répétitif de cette activité constitue un risque important d'erreurs.
Hors la cohérence entre la documentation produite et le système décrit est primordiale, car l'objectif est ici, de fournir des documents de référence. Des erreurs dans ce domaine peuvent avoir des conséquences très importantes et le cas échéant, être à l'origine de litiges entre les parties.

Pour le cas de la documentation des bases de données, ces différents aspects sont encore renforcés de par les volumes à traiter. Dans ce contexte, l'automatisation du processus nous apparaît donc comme nécessaire, car de tels volumes de documentation ne peuvent être créés et tenus à jour de manière manuelle.

Avec les technologies basées sur les compilateurs et les parseurs, il nous est possible de proposer des automates capables de parcourir les codes sources et d'en extraire l'information recherchée. Outre la réduction des coûts, cette approche présente l'avantage de fournir des documents à jour rapidement, tout en présentant une certaine garantie de fidélité, car générés avec une information directement extraite du code exploité. L'automatisation permet donc de réduire les risques d'incohérence.

Toutefois, si la génération automatique apparaît comme une approche souhaitable pour produire et maintenir de grands volumes de documentation, tout ne doit pas être automatisé, en effet, le risque de générer des documents peu utilisables existe. La génération automatique constitue donc une réponse intéressante au problème si le processus reste bien contrôlé par les utilisateurs, notamment par un paramétrage approprié.

2.3.6. La mise en œuvre de standards

L'objectif de l'activité de documentation est l'échange de l'information. L'adoption de formats standards comme support, notamment pour la publication, présente donc des avantages évidents. D'une part, l'utilisation de formats reconnus et utilisés dans la pratique facilite la diffusion et l'adoption des documents produits. D'autre part, l'adoption d'un format normalisé est une condition de départ pour l'échange, car ces formats garantissent une représentation identique de l'information qui est nécessaire pour permettre aux systèmes de communiquer [Marcoux 94].

Il est également intéressant de remarquer que la mise en œuvre de formats génériques réduit les besoins de développement, car d'une part, ces formats sont généralement accompagnés d'outils et de fonctionnalités que l'on peut avantageusement réutiliser et que d'autre part, pour le déploiement de l'outil de documentation, la prise en compte par les grands acteurs industriels de la technologie utilisée est un facteur essentiel pour la portabilité de la solution sur les plates-formes existantes.

2.4. Conclusions de l'étude

Au terme de cette étude du besoin de documentation, il nous paraît utile de dégager les quelques points suivants :

- Comme pressenti, il apparaît bien que la documentation constitue un besoin essentiel pour les développeurs de base de données, ceci est confirmé par le développement des activités de rétro-ingénierie.

- En matière de documentation, l'activité de développement des bases de données présente des besoins spécifiques, qui sont de documenter les schémas et les transformations de schémas.

- Cette activité est également confrontée aux volumes importants de documentation à produire qui sont liés aux tailles importantes des systèmes actuels.

- La qualité d'une documentation doit être jugée sur son utilisation effective et donc la conception d'un outil de documentaire doit être guidée par les besoins des utilisateurs.

- L'information relative à ces systèmes d'information étant située à la fois dans les codes sources et chez les acteurs ayant participé à leurs développements, le système documentaire proposé doit pouvoir offrir des fonctionnalités permettant d'extraire et de traiter ces deux types d'information.

- L'automatisation de la génération de la documentation apparaît comme nécessaire pour permettre sa production à des coûts et dans des délais raisonnables.

Ces conclusions guideront nos choix durant le développement de la solution proposée pour l'outil de documentation de DB-Main.

3. L'informatique documentaire

Ce chapitre, présente un rapide état de l'art des techniques de l'informatique documentaire. Après avoir présenté les standards technologiques et les normes, nous nous intéresserons aux avantages des formats de documents structurés et de l'hypertexte pour l'activité documentaire.

3.1. Les technologies de l'informatique documentaire

3.1.1. Les standards technologiques

Dans le domaine de la documentation technique et parmi les technologies de l'informatique documentaire quelques grands standards techniques semblent avoir émergé. Nous retrouvons notamment :

3.1.1.1. TEI

TEI (Text Encoding Initiative) est un standard bien établi dans le domaine de l'édition et des bibliothèques. L'objectif de TEI est de proposer des modèles très expressifs et puissants pour pouvoir représenter tout type de texte ainsi que pour assurer le traitement de l'information. Cette approche a pour conséquence que les solutions proposées sont assez complexes. Les technologies développées pour TEI s'appuient principalement sur SGML pour les règles de codage bien qu'elles s'ouvrent actuellement à XML.

3.1.1.2. DocBook

DocBook est un standard de la documentation technique, qui s'appuie également sur SGML et XML. Ce standard initialement développé par Norman Walsh pour Hal Computer Systems est aujourd'hui supporté par le consortium OASIS dont l'objectif est de soutenir et diffuser les standards professionnels ouverts. DocBook fait aujourd'hui l'objet de nombreux développements notamment pour la mise en ligne de la documentation et ce format est fortement utilisé dans le secteur informatique (Editions O'Reilly, Sun, Novel, IBM Linux Technologie Center, etc.).

3.1.1.3. DITA

La Darwin Information Typing Architecture de chez IBM est le format XML pour la documentation technique proposé par ce grand acteur. DITA réutilise des concepts assez similaires à ceux mis en œuvre par DocBook.

3.1.2. Les produits commerciaux

A côté de ces standards technologiques, nous trouvons également un grand nombre de produits commerciaux dédiés à la documentation dont les objectifs et les orientations sont très variés. Cette variété s'étend des applications permettant à la génération automatique de documentation de programme : Javadoc, Doc-o-Matic, etc. jusqu'aux solutions complètes de Knowledge Management et de portail d'entreprise.

3.1.3. Tendances générales

De ces standards technologiques et de ces produits commerciaux, il ressort assez clairement 2 tendances générales qui sont :

1. La mise en œuvre d'une information de type structurée. Comme l'indique le tableau ci-dessous, les standards technologiques font pour leurs parts appelle aux formats SGML et XML.

Standard technologique	*Format normalisé de document structuré utilisé*
TEI	SGML et XML
DocBook	SGML et XML
DITA	XML

Figure 1 Formats normalisés mis en œuvre dans les standards technologiques

2. L'utilisation de l'hypertexte comme support de publication. Parmi les formats de publication offerts par les systèmes examinés, certains formats reviennent régulièrement, nous avons notamment les documents de type : PDF, PS, RTF, TEX, etc. qui sont souvent disponibles. Pour sa part, le format HTML est lui systématiquement proposé comme support pour les documents produits.

3.2. L'information de type structurée

3.2.1. Définition

Par information structurée, nous entendons une information qui répond à un ensemble de règles qui permettent à chaque élément d'information d'être :

1. Clairement identifié, généralement par l'utilisation d'un balisage. Il s'agit ici d'un balisage de type descriptif utilisé pour délimiter les objets conceptuels : des mots, des phrases, des types d'entité, des attributs, etc. et non d'un balisage procédural destiné lui à simplement indiquer la mise en forme à appliquer à l'élément [Marcoux 94].

2. Précisément localisé par sa position dans une arborescence qui le situe par rapport aux autres éléments d'information (ses ascendants, ses descendants, ses semblables, etc.). Cette arborescence introduit une structure hiérarchique aux documents qui s'apparente au modèle de données en arbres et qui sera fondamentale pour les traitements de ces documents.

Sur bases de ces critères, et à titre de comparaison, nous pouvons qualifier :

- Les documents textes ou de type traitement de texte comme non structurés.
- Les documents HTML comme peu structurés.
- Les documents SGML et XML comme fortement structurés.

3.2.2. Concepts et avantages des formats de documents structurés

Dans le cadre d'une activité documentaire, les concepts et les avantages apportés par ces formats sont les suivants :

- Une Source unique pour l'information.

  - **Avantage :** La factorisation du travail que permet une source commune de l'information est un avantage important qui se traduit par des gains de temps et de coûts [Marcoux 94]. Un seul document XML peut être utilisé pour produire de manière automatique différents formats de publication.

- Le principe de séparation entre le contenu et la présentation. Avec les formats de documents structurés, une différenciation claire entre les aspects sémantiques et les aspects de formatage est introduite.

  - **Avantage :** Sur un plan organisationnel, la séparation de l'activité rédactionnelle de l'activité de présentation/publication permet aux équipes en charge de ces activités de travailler indépendamment l'une de l'autre.

- La structuration en arborescence.

  - **Avantage :** Ce modèle de données autorise certaines formes de traitement et d'automatisation dont notamment la validation, les transformations et les chaînes de traitements.

- L'utilisation de métadonnées.

  - **Avantage :** L'introduction d'informations additionnelles qui caractérisent l'information ou le document lui-même est particulièrement utile. Les métadonnées peuvent servir notamment pour l'indexation, l'échange de données, la recherche documentaire, etc.

Les formats de documents structurés peuvent également présenter certains inconvénients. Tout d'abord, ils sont plus complexes à mettre en œuvre, c'est donc un risque supplémentaire dont il faut tenir compte, ensuite, ils occasionnent certains coûts notamment pour l'acquisition et l'apprentissage des nouveaux outils.

3.2.3. Les formats normalisés de documents structurés

Nous présentons ici les principaux formats faisant l'objet de normalisation internationale. Nous abordons dans un premier temps, les normes ISO puis, dans un second temps, les recommandations du W3C. Pour chaque partie, les normes sont présentées dans l'ordre chronologique de leur parution.

Les normes ISO

3.2.3.1. La norme SGML (1986)

La norme SGML pour Standard Generalized Markup Language fut la première norme de document structuré proposée. Elle est issue des travaux de Charles Goldfarb (IBM) qui introduit dès 1969 l'utilisation d'un balisage généralisé pour structurer les documents. Normalisé par l'ISO (ISO 8879), ce format fut adopté et diffusé via des projets d'envergure tels que le projet de support logistique CALS du département de la défense américaine. Le projet TEI a également contribué à la diffusion de cette norme en particulier dans le monde de l'édition où le format SGML a pris une importance considérable à travers notamment un grand nombre de produits commerciaux.
Une caractéristique intéressante de SGML est évidemment l'ouverture apportée par le balisage généralisé qui ne se limite pas à proposer un balisage descriptif ou procédural, mais qui permet de définir de véritables langages via les DTD (Document Type Definition) . TEI et HTML sont tous deux des DTD publiques de SGML.
Une autre caractéristique importante de SGML est que la séparation introduite entre les aspects contenu et la mise en forme du document, est formalisée par la norme. En effet, le formatage est défini séparément à travers les feuilles de style DSSSL (Document Style Semantics and Specification Language).

3.2.3.2. La norme ODA (1989)

Issue des travaux de l'European Computer Manufacturer Association, l'Office Document Architecture ODA a été normalisé par l'ISO en 1989 (ISO 8613). La particularité de ce format est de proposer une approche orientée objet pour structurer les documents et d'intégrer directement la structure logique et la structure physique de manière à obtenir un format strict, le balisage utilisé ici est codé. ODA est un format clairement orienté pour l'échange de documents et il introduit très tôt la possibilité d'inclure des éléments multimédias. Bien que supporté dés le début des années 90 par un consortium comprenant des sociétés telles qu'IBM, DEC, Olivetti son développement semble ne pas avoir rencontré le même succès que celui obtenu par SGML.

3.2.3.3. La norme HyTime (1992)

La norme ISO (ISO 10744) relative à l'Hypermedia Time-based Document structuring Language a été adoptée en 1992. Cette dernière peut être vue comme une extension de la norme SGML permettant l'introduction de composants multimédia et la prise en compte de l'hypertexte. Pour ce faire, HyTime fait appel à un ensemble de balises spécifiques permettant d'inclure les éléments multimédias, mais aussi les liens hypertextes. A ce jour, peu de produits commerciaux ont été développés pour ce format.

Les recommandations du W3C

Parallèlement aux Normes ISO, les recommandations du W3C (World Wide Web Consortium) ont également contribué de manière importante à la normalisation et plus en encore à la diffusion des formats de documents structurés.

3.2.3.4. La recommandation HTML (1989)

L'Hypertext Markup Language est à l'évidence le format le plus largement diffusé, ceci est sans doute lié à sa simplicité et à sa grande facilité d'utilisation qui en ont fait le format incontournable de l'Internet et des Intranets de sociétés. Comme mentionné précédemment, HTML est une application de SGML, la DTD HTML définit un ensemble de balises spécifiquement orientées pour la publication de documents en ligne.
Cette approche à conduit à la définition d'un balisage descriptif et procédural fortement axé sur les aspects de formatages des pages et qui a pour conséquence certaines limitations dans l'exploitation qu'il est possible de faire des documents HTML. Une autre faiblesse de ce format est le mélange de structure logique, de structure physique et même de code qui est autorisé, ce mélange peut conduire à diverses incohérences, comme par exemple, la rupture de la structure hiérarchique.
Par contre, l'apport majeur de l'HTML par rapport à SGML, est l'utilisation et la normalisation des liens hypertextes et du mécanisme d'adressage qui l'accompagne. Bien que les principes de l'hypertexte soit bien antérieurs au développement de ce format, le terme « hypertext » apparaît déjà en 1965 dans les travaux de Theodor Holm Nelson et les concepts sont attribués à Vannevar Bush 1945 [Ben Romdhan 01], il n'en reste pas moins que c'est bien HTML qui a contribué à rendre la navigation hypertexte réellement populaire. On peut d'ailleurs noter qu'une des conséquences intéressantes de la forte diffusion de ce format est le développement et la généralisation de navigateurs Web puissants sur les postes de travail des informaticiens.

3.2.3.5. La recommandation XML (1998)

La recommandation XML (eXtensible Markup Language) est en fait un sous-ensemble de SGML, nous retrouvons donc des caractéristiques de ce langage telles que le traitement du contenu effectué indépendamment de la mise en page ainsi que l'utilisation des DTD.
L'objectif poursuivi par le W3C lors de la parution de XML en 1998 était de proposer un format d'échange de documents bénéficiant de la puissance de description du balisage généralisé tout en conservant les qualités de simplicité qui avait fait le succès d'HTML.
Le souhait était donc d'établir un standard d'échange de données sur le Web avec une démarche similaire à celle qui avait permis à HTML de s'imposer comme standard d'échange de documents.

XML est une recommandation récente qui ne s'est pas encore généralisée d'une manière aussi importante qu'HTML, pourtant ce format est l'objet de nombreux développements comme en témoigne le nombre important de DTD spécialisées qui sont apparues récemment, et ce, dans des domaines très variés : MathML Mathématical Markup Language pour l'échange de documents comprenant des formules, CML dans le domaine de la chimie, AIML pour l'astronomie, BSML pour la bio-informatique, MusicML, etc..

Un autre signe de l'intérêt porté à XML est le nombre d'outils et d'applications compatibles ou spécifiquement développés pour ce format. On peut par exemple citer ici, la compatibilité de XML avec les feuilles de style CSS (Cascading Style Sheets), le langage de transformation XSLT (eXtensible Stylesheet Language), les extensions à l'hypertexte Xlink et XLL (Extensible Linking Language), etc..

XML versus SGML

Lorsque l'on compare ces deux formats, on parle souvent de XML comme d'une simplification de SGML. En effet, SGML comprend un très grand nombre de possibilités optionnelles qui ne sont pas toujours nécessaires, mais qui introduisent une certaine lourdeur. XML a pour sa part tenté de conserver les avantages essentiels de son aîné, tout en limitant la complexité et en laissant aux utilisateurs de la norme, le soin de développer les options dont ils ont besoin. Dans la pratique, il s'avère également qu'il est plus aisé d'utiliser XML que SGML qui est plus difficile à maîtriser. Le langage de transformation XSLT est par exemple bien plus simple que DSSSL qui est le langage de transformation de SGML. Il faut enfin noter que SGML ne prévoie pas nativement l'hypertexte.

3.2.3.6. La recommandation XHTML (2000)

XHTML est une extension des spécifications de la recommandation HTML 4.
Les motivations du W3C pour l'introduction de ce nouveau standard sont bien entendu liées aux limitations d'HTML dont l'obsolescence est un frein au développement d'applications Web réellement performantes, mais aussi, d'établir les bases d'un langage modulaire et extensible permettant l'intégration progressive des concepts de XML dans ces applications. Pour ce faire, une redéfinition du balisage de l'HTML sous la forme d'une DTD XML a été proposée, aboutissant ainsi à des documents offrant l'avantage d'une compatibilité dans les 2 sens. Un document XHTML est donc un document XML bien formé, mais également un document compatible avec HTML 4 et donc avec les nombreuses applications traitant ce format, dont les navigateurs. Notons également que des outils de conversion automatique entre les versions antérieures d'HTML et XHTML sont proposés et que les applications utilisant des scripts et des applets sont correctement prises en compte par ce nouveau format.

3.3. L'hypertexte

3.3.1. Définition et principes

Parmi les définitions existantes, nous reprenons ici la définition de référence proposée par Conklin.

« Une base de données textuelles, visuelles, graphiques, sonores où chaque îlot d'information est appelé nœud ou cadre ; l'ordinateur établit des liens potentiels entre ces nœuds et peut ainsi créer un mouvement rapide dans cette masse d'informations ; une interface ou un monde de présentation visuelle permet l'interaction entre l'usager et l'hypermédia ». [Conklin 87]

Cette définition nous permet d'introduire les principes de base de l'hypertexte. Il s'agit donc, de documents incluant un ensemble de liens référencants des parties d'informations appelées nœud, établissant ainsi des liens sémantiques entre des documents ou des parties de document. L'intérêt de cette approche est d'offrir au lecteur une grande liberté de consultation, en lui permettant d'accéder de manière plus directe à l'information.

Dans sa généralisation sur le réseau Internet, la technologie hypertexte s'est également enrichie d'outils particulièrement intéressants pour la recherche et le filtrage d'informations. Nous avons notamment :

- Les moteurs de recherches capables d'indexer des millions de documents.
- Les annuaires communautaires.
- Les outils d'indexation automatiques qui utilisent des métadonnées spécialisées. A titre d'exemple, le Dublin Core qui est actuellement en cours de normalisation par l'ISO, propose une quinzaine de métadonnées destinées à faciliter le travail d'indexation des documents du Web.
- Plus récemment, sont aussi apparus des agents de recherche dont l'objectif est de butiner automatiquement le réseau à la recherche d'informations ciblées.

3.3.2. La problématique de la navigation hypertextuelle

Malgré la puissance des concepts et des outils de recherche disponibles, la consultation de document hypertexte demeure problématique [Ben Romdhane 01]. Cette difficulté est liée d'une part, à la désorientation occasionnée par une navigation rapide au travers de grandes masses de documents, après plusieurs « sauts » dans l'hypertexte, le lecteur se retrouve perdu sur des pages de documents qu'il n'avait pas nécessairement prévu de consulter. Et d'autre part, à une surcharge cognitive qui résulte du nombre d'actions simultanées en cours : lecture de la page, prise de décision sur les liens à suivre, mémorisation des chemins parcourus, introduction des requêtes dans les moteurs de recherche, évaluation de la pertinence de l'information trouvée, etc..

Cette problématique est importante et nous concerne directement étant donné les volumes des documents à produire dans le cadre de la documentation de grands schémas de bases de données.

3.3.3. L'intérêt des liens hypertextes structurants

La prise en compte de la problématique exposée ci-dessus est essentielle, la facilité d'usages et la compréhension des documents produits sont des facteurs importants dans l'acceptation d'un outil documentaire par les utilisateurs.

Une réponse intéressante à ce problème consiste à accompagner le document d'une structuration forte qui peut être obtenue en faisant appel à de nombreux liens de type structurel [Scott 93]. Cette solution consiste à inclure dans les pages HTML, en plus des liens référentiels qui orientent le lecteur directement vers l'information, des liens structurels qui vont lui permettre de comprendre, de manière intuitive, la structure et le découpage du document. Ces liens permettront, par exemple, de circuler dans une hiérarchie de sections et de sous-sections, d'aller rapidement vers un point de passage clef du document (la table des matières, l'index, le chapitre suivant, etc.), ou encore tout simplement, de retourner au début du document.

Cette démarche de structuration peut être portée plus loin encore, en proposant à l'utilisateur une personnalisation complète de la structure du document, lui permettant ainsi de l'adapter à son besoin et à ses souhaits en terme d'ergonomie [Scott 93].

3.4. Conclusions

Il ressort des éléments présentés dans ce chapitre que dans une optique de génération de documentation, les formats HTML et XML offrent de réels intérêts.

En effet, d'une part, sur le plan de la diffusion et de la visualisation de l'information, les publications de type hypertexte, à base de pages HTML et faisant appel aux navigateurs installés sur les postes de travail, conviennent bien à la consultation de documentation. Ce mode de consultation est en outre familier pour le public de professionnels de l'informatique visé dans notre travail. Et d'autre part, le format XML constitue aujourd'hui le format de référence pour le développement d'applications mettant en œuvre une information de type structurée.

Toutefois, le choix d'un bon format de publication ne suffit pas à assurer le succès d'une application documentaire en particulier dans le cadre de production de documents volumineux comme c'est notre cas. La facilité de lecture et de prise en main des documents constituera sans doute un facteur essentiel pour l'adoption de la solution par les utilisateurs. C'est la raison pour laquelle, nous proposons de retenir une solution qui permette la génération de publications bien structurées, notamment par l'introduction de liens structurels dans les documents produits.

4. Solutions retenues pour la génération de la documentation de DB-Main

Le second objectif de ce mémoire consiste à proposer un générateur de documentation de type hypertexte pour l'atelier DB-Main. Dans ce chapitre, après avoir établi l'analyse des besoins de l'outil et étudiés les éléments existants dans l'atelier, nous présentons les solutions retenues pour remplir cet objectif ainsi que les justifications de ces choix.

4.1. Analyse des besoins

L'analyse des besoins présentée ici est issue des souhaits exprimés par les membres de l'équipe du LIDB ainsi que des éléments mis en évidence lors de l'étude du besoin de documentation.

4.1.1. Les besoins fonctionnels

1. La première exigence auquelle doit satisfaire la solution proposée est de répondre aux besoins de base de documentation pour les bases de données, qui sont d'une part, de documenter les schémas qui spécifient le système et d'autre part, de présenter les documents textuels existants.

   La documentation des schémas
   Il s'agit de fournir une documentation complète et précise des composants de ces schémas. Dans le cadre de DB-Main, nous avons 2 grandes classes de schéma, qui sont d'une part, les schémas Data et d'autre part, les schémas Process.
   - Les schémas Data sont les schémas qui modélisent le domaine traité aux différentes étapes du processus de développement, nous retrouvons :
     - Les schémas conceptuels
     - Les schémas logiques
     - Les schémas physiques
   - Les schémas Process ont eux pour fonction de modéliser les programmes et les procédures qui accompagnent la DB. Il peut s'agir par exemple, de modéliser le comportement de triggers ou de scripts qui assurent l'intégrité de la DB.

   Le tableau ci-dessous présente une synthèse des composants à documenter pour chaque type de schéma.

Type Schéma	*Composants à documenter*
Conceptuel	Types d'entité, types d'association, attributs, clusters, rôles, groupes, domaines définis par l'utilisateur, descriptions et notes.
Logique	Tables, colonnes, clefs, contraintes, collections, descriptions et notes.
Physique	Tables, colonnes, clefs, contraintes, collections, descriptions et notes.
Process	Unités de processus, données internes et externes, relations, descriptions et notes.

Figure 2 Synthèse des composants à documenter

La présentation des documents textuels
Il s'agit ici de permettre simplement au lecteur de consulter ces documents, de préférence au moyen la même interface de consultation que celle proposée pour la documentation des schémas.

2. La seconde exigence est de pouvoir prendre en charge de grands schémas de base de données, pouvant compter jusqu'à 1000 éléments (types d'entité, types d'association, tables). L'outil proposé doit donc permettre de produire de gros volumes de documentation. Cette contrainte souligne l'importance de fournir une solution utilisable par des professionnels qui sont dans la pratique confrontés à des systèmes de taille conséquente.

3. L'outil doit également permettre de documenter les opérations de transformation effectuées sur les schémas. Cette exigence implique de faire appel à des fonctionnalités de traçabilité forward/backward pour obtenir cette information très spécifique.

4. La documentation produite sera générée de manière automatique. Dans notre contexte, l'automatisation du processus de génération de la documentation est nécessaire pour permettre de traiter les grands volumes dont il est question.

5. La documentation sera produite sous la forme d'hypertexte et les documents hypertextes générés comprendront des liens de type structurel facilitant la navigation. Le format de publication proposé respectera les recommandations HTML 4.01 du W3C permettant leurs consultations avec les navigateurs standards.

6. L'outil de documentation sera paramétrable afin de générer des rapports sur mesure. L'accent sera mis sur la sélection des éléments qui doivent être documentés, afin de répondre au mieux aux besoins des lecteurs, sans pour autant produire une documentation excessive.

4.1.2. Les besoins non fonctionnels

1. Afin de proposer un générateur de documentation bien intégré à DB-Main, on souhaite que l'implémentation de la solution prenne la forme d'un plug-in Voyager qui pourra être appelé directement depuis l'interface de travail de l'outil Case.

2. L'architecture de la solution proposera une bonne modularité propre à faciliter la maintenance de l'outil et permettant son évolution vers de futurs besoins, par exemple pour l'extension de la documentation à d'autres parties des projets DB-Main.

3. L'utilisation de standards technologiques et de formats normalisés est souhaitée, cette démarche devrait permettre de limiter les besoins de développements spécifiques.

4.2. Etude des éléments existants

Nous étudions ici les composants de l'atelier logiciel DB-Main avec comme objectif de mettre en évidence les fonctionnalités pouvant être impliquées dans la construction de l'outil de documentation.

4.2.1. L'atelier DB-Main

L'atelier logiciel DB-Main, aujourd'hui dans sa version 7, est le résultat d'un programme de recherche de longue haleine conduit depuis 1993 au sein du LIBD. Destiné à fournir un support pour les activités d'ingénierie logicielle relatives aux bases de données, cet outil CASE peut bien sûr être mis en œuvre pour le design de nouvelles bases de données, mais aussi, pour des activités de ré-ingénierie et de rétro-ingénierie dont nous avons souligné le développement actuel. Disposant de nombreuses fonctionnalités, l'atelier logiciel constitue donc un outil performant, dont l'usage s'adresse aux professionnels de l'activité.

Pour atteindre ses objectifs, DB-Main s'appuie sur 5 principes architecturaux [Hainaut 99] qui sont :

1. Le repository de DB-Main qui reprend dans une structure de données unique toutes les spécifications relatives à un projet.
2. Un ensemble extensible d'outils de traitement (plug-in).
3. Un processus d'ingénierie basé sur les transformations de schémas.
4. Des guides méthodologiques interactifs.
5. Un modèle de développement ouvert permettant notamment l'introduction de nouvelles fonctionnalités et l'extension du méta-schéma de la structure de données.

4.2.2. Exploitation du Repository géré par DB-Main

Le repository est la structure de données qui assure stockage des composants des projets DB-Main. Il s'agit d'un ensemble d'objets et de relations qui répondent à un schéma de définition générique et qui permettent de spécifier complètement le système développé.
C'est du repository que seront extraites les informations qui constitueront le contenu de la documentation.

4.2.3. Le langage de développement VOYAGER

Voyager est un langage programmation de type impératif spécifiquement développé pour exploiter le repository DB-Main et étendre les fonctionnalités de l'atelier [Englebert 02]. Il offre évidemment d'intéressantes perspectives puisque nous disposons avec ce langage d'un outil permettant l'accès direct aux données des projets.

Par rapport à l'environnement de travail, les compilateurs Voyager permettent de générer des plug-ins (fichier .oxo) qui peuvent être appelés directement depuis l'interface de l'outil CASE.

4.2.4. Exploitation des fonctionnalités existantes dans l'atelier.

Parmi les nombreuses fonctionnalités disponibles, certaines nous intéressent particulièrement :

4.2.4.1. Les descriptions sémantiques et les notes

Les descriptions et les notes ont pour but de saisir de l'information associée aux composants d'un projet, par exemple, la sémantique d'un type d'entité d'un schéma conceptuel, une explication sur un attribut, un commentaire sur une table d'un schéma logique, etc.. Il s'agit donc de saisies textuelles qui permettent de lier l'information directement aux objets auxquels elles se réfèrent. Par ce moyen, DB-Main propose une fonctionnalité très intéressante, car elle permet la capture des connaissances des analystes et des développeurs via une interface bien intégrée à l'outil. Les commentaires introduits au moyen de cette fonctionnalité, fournissent évidemment des indications précieuses pour la compréhension du système et la collecte de cette information est donc nécessaire, car elle constituera une part importante de la documentation à produire.

4.2.4.2. La fonction Mark

Cette fonction permet d'effectuer une sélection d'objets par marquage de ceux-ci. Cette sélection peut être faite au niveau des objets d'un schéma, mais il est aussi possible de sélectionner les schémas eux-mêmes avec l'ensemble de leurs composants. La fonction Mark pourra être utilisée pour déterminer de manière précise quels sont les éléments à documenter.

4.2.4.3. La fonction Trace

Le journal intégré à DB-Main peut enregistrer toutes les actions du processus d'ingénierie appliqué et notamment les transformations de schéma utilisées lors du développement. Chaque manipulation sur les composants du projet est consignée dans un fichier de trace permettant des retours en arrière de type « Undo » ou encore de rejouer toute une séquence d'actions. Cette fonction permet donc bien de collecter les éléments de la traçabilité Forward/Backward qui permettent de documenter les transformations de schéma. L'utilisation de cette fonction sera donc envisagée comme solution possible pour ce besoin.

4.2.4.4. Les métas propriétés

Chaque type objet du repository peut-être accompagné d'attributs d'information génériques, ce sont les métas propriétés de l'objet. Une caractéristique originale de l'outil CASE DB-Main est d'autoriser l'extension de ces métas propriétés offrant ainsi la possibilité d'introduire, selon nos besoins, de l'information additionnelle sur les objets. Les métas propriétés pourraient donc constituer une nouvelle source d'information très intéressante.

Remarque

L'atelier DB-Main comprend déjà une fonction de documentation qui permet de générer automatiquement des rapports au format RTF (document Word). Ces rapports sont générés sur base d'un template fixe et sont jugés trop simples et trop statiques pour constituer un outil documentaire. De plus, sur le plan de la facilité d'usage, ils ne sont pas bien adaptés pour produire la documentation de grands schémas.

4.3. Solutions retenues

Sur base des éléments relevés ci-dessus, et en tenant compte des solutions techniques rencontrées lors de notre étude de l'informatique documentaire, nous proposons de retenir le standard de documentation technique **DocBook dans sa version XML** pour le développement du générateur de documentation.

4.3.1. DocBook Version 4.2

Ce standard de la documentation technique semble pouvoir bien répondre à nos besoins pour les raisons suivantes :

- DocBook est un format documentaire simple et robuste qui a été éprouvé sur des chaînes de production de documents lourds. Son utilisation par de grands noms du monde de l'édition (O'Reilly) ou de l'informatique (Sun, Novel, IBM) constitue sans doute un argument en faveur de son efficacité. Ceci devrait nous permettre de rencontrer la contrainte de grands volumes de documentation à produire.

- DocBook est également un standard bien établi, utilisé et supporté par une large communauté, il existe une abondante documentation sur ce standard et ses extensions. C'est aussi une solution ouverte qui s'appuie elle-même sur les standards et les normes (SGML, XML, HTML) et qui est indépendante des acteurs commerciaux.

- Un autre avantage important de DocBook pour l'outil de documentation envisagé est l'utilisation de la métaphore familière du livre. Nous pensons en effet, que les concepts bien connus que sous-tend le modèle du livre tel que les chapitres, les sections, la table des matières, etc., peuvent faciliter la prise en main des documents proposés.

- Dans sa version XML, DocBook permet avec différents outils de produire plusieurs types de formats de publication dont : HTML, PDF, RTF, TEX, LaTEX et PS. Nous retrouvons donc parmi les formats proposés le format HTML souhaité, de plus ce format peut-être ici décliné dans différentes variantes :

    - **Single HTML** : Document HTML en une seule page. Indiqué pour les petites documentations.

    - **Multi HTML** : Documentation présentée au travers d'un ensemble de pages HTML. Multi HTML qui est le format de documentation de base de DocBook utilise ici pleinement la métaphore du livre. Les notions de chapitres, de sections, de paragraphes, etc. sont comme nous le souhaitions obtenues par l'introduction de nombreux liens de navigation qui mettent clairement en évidence la structure du document. Dans cette variante, DocBook en propose également la génération automatique de tables des matières et d'index avec ici encore des liens vers les différentes partie du document.

- **Help HTML** : Il s'agit du format d'aide bien connu qui présente l'avantage de rassembler sous un seul document (fichier .chm) les nombreux fichiers HTML générés et de les conserver dans un format comprimé. Le format Help HTML offre également un ensemble de fonctionnalités très utiles qui facilitent la consultation dont un outil de recherche plein texte, une mise en forme élaborée de la table des matières et de l'indexation, une fonctionnalité de gestion des favorites, etc. Les fichiers .chm doivent être visualisés au travers du navigateur d'aide de Microsoft qui est fourni en standard avec l'OS Windows.

- Enfin, il nous semble également intéressant de souligner les 2 points suivants :

1. D'une part, les technologies liées à DocBook sont « libres », dans le sens où les DTD et les feuilles de style sont publiées sous une licence « Open source ».
   Ceci implique :
   - Qu'elles sont utilisables et distribuables gratuitement et sans limitation !
   - Qu'il n'y a pas de risque de voir ces technologies disparaître avec les sociétés commerciales qui les auraient développées, contrairement aux formats propriétaires !
   - Que les codes sources sont disponibles et qu'ils peuvent être modifiés pour répondre au mieux aux besoins rencontrés.

2. D'autre part, la documentation produite avec le modèle DocBook devrait pouvoir évoluer rapidement vers un système d'information de type « portail Web ». Ce format est en effet maintenant supporté, de manière native, par l'infrastructure de publication Web Forrest qui est une extension du système Apache.

Avec le support du consortium OASIS, DocBook nous apparaît donc comme un standard professionnel ouvert, bien adapté à nos besoins.

4.3.2. Le format de document structuré XML

Pour la génération de la documentation de bases de données, les avantages du format XML sont importants. Nous avons déjà souligné qu'actuellement, XML est le format de référence pour la mise en œuvre de l'information structurée, mais nous pouvons aussi mettre en avant d'autres avantages, notamment :

- Un mécanisme structurant bien adapté à la documentation technique où la maîtrise de la structure du document est importante [Scott 93]. Ce mécanisme est en outre, considéré comme particulièrement indiqué pour la représentation de schémas de base de données. Nous pouvons citer ici Hartmann, Huang et Tilley :

  « The hierarchical nature of XML documents provides useful structuring mechanisms for program artifacts. For example, XML constructs can be used to represent both database schemas and database elements that capture important source-level information from software program. » [Hartmann 01]

  Ceci constitue évidemment un avantage déterminant pour rencontrer notre besoin de base qui est de documenter les schémas et leurs structures particulières.

- La possibilité d'intégrer différentes sources d'information. Comme indiqué dans l'analyse des besoins, l'intégration de documents de différentes natures est un objectif. Les capacités d'intégration de XML nous permettront d'inclure directement les fichiers textuels dans la documentation.

- La vérification de la consistance des informations peut être obtenue par des opérations de validation par rapport aux DTD. Ces dernières permettent de formaliser de manière précise les règles de conformité d'un document.

- L'existence de nombreux outils (éditeurs, parseurs) constitue bien sûr un atout. Nous avons notamment plusieurs types de parseurs prêts à l'emploi et bien documentés qui permettent le développement rapide d'applications dans différents langages.

- La portabilité de XML sur les différentes plates-formes constitue aussi un avantage appréciable. Ce facteur ainsi que l'engouement des grands industriels de l'informatique pour ce format nous laissent entrevoir l'adoption de XML comme un standard pérenne.

- Enfin, dans la perspective du développement d'un outil documentaire complet, la disponibilité de bases de données, acceptant le format XML comme source d'information, permet d'envisager la création rapide d'un système de gestion de la documentation produite. En effet, de nombreux fournisseurs proposent dès aujourd'hui des solutions d'interfaçage de leurs SGBD permettant soit, l'utilisation directe de documents XML comme source d'information (DB au format XML natif), soit des modules de traitement permettant la conversion des documents XML vers le format d'exploitation de leurs systèmes.

  Ce dernier point est important, car il offre une approche intéressante pour répondre au problème de la gestion de la documentation.

4.3.3. Automatisation par transformations XSLT

Le format XML avec son modèle de données en arborescence et ses techniques de transformation XSLT est bien adapté pour les traitements automatiques. Avec ces techniques, ce format peut être facilement converti vers les différents types de publication. Pour la navigation, le format XML est directement transformé en HTML et pour l'impression, des outils nous permettent une conversion automatique vers les formats standards : PDF, PS, RTF, TEX, etc.

Dans le cadre d'une activité documentaire, la chaîne de traitement des documents XML prend la forme suivante :

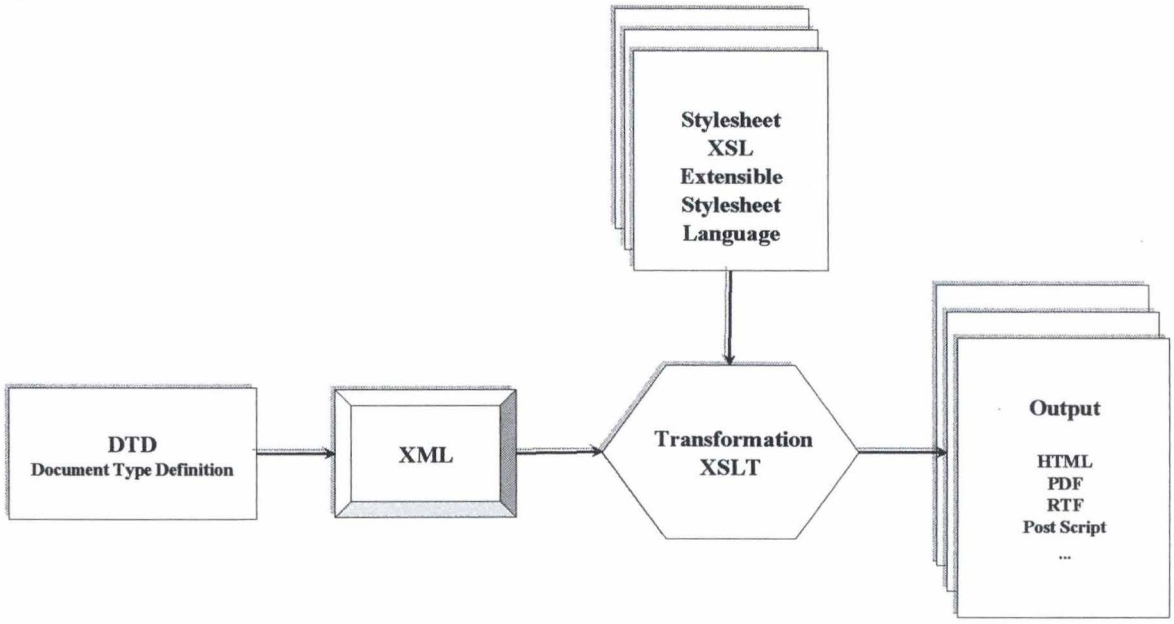

Figure 3 Chaîne de traitements XML

4.3.4. Développement d'un plug-in Voyager

Le développement d'un plug-in Voyager présente plusieurs avantages :

- Il permet tout d'abord d'accéder aux données du repository de manière efficace via une librairie de fonctions spécifiquement développées pour cet usage. Voyager permet notamment de parcourir la structure particulière du repository pour en extraire les informations souhaitées.

- Les plug-ins Voyager offrent naturellement une intégration poussée à l'outil CASE, l'interface de l'outil autorisant le chargement et l'exécution directe de ces programmes.

- Enfin, Voyager est un langage de programmation de type impératif qui après compilation des sources, permet une exécution rapide des traitements demandés.

4.4. Principes de séparation contenu/présentation et présentation/publication

4.4.1. Formats et types de fichier mis en oeuvre

Nous présentons dans le schéma suivant, les formats de documents utilisés dans la solution. Ce schéma met bien en évidence, la séparation introduite entre les parties extraction du contenu, présentation du document et publication au format souhaité.

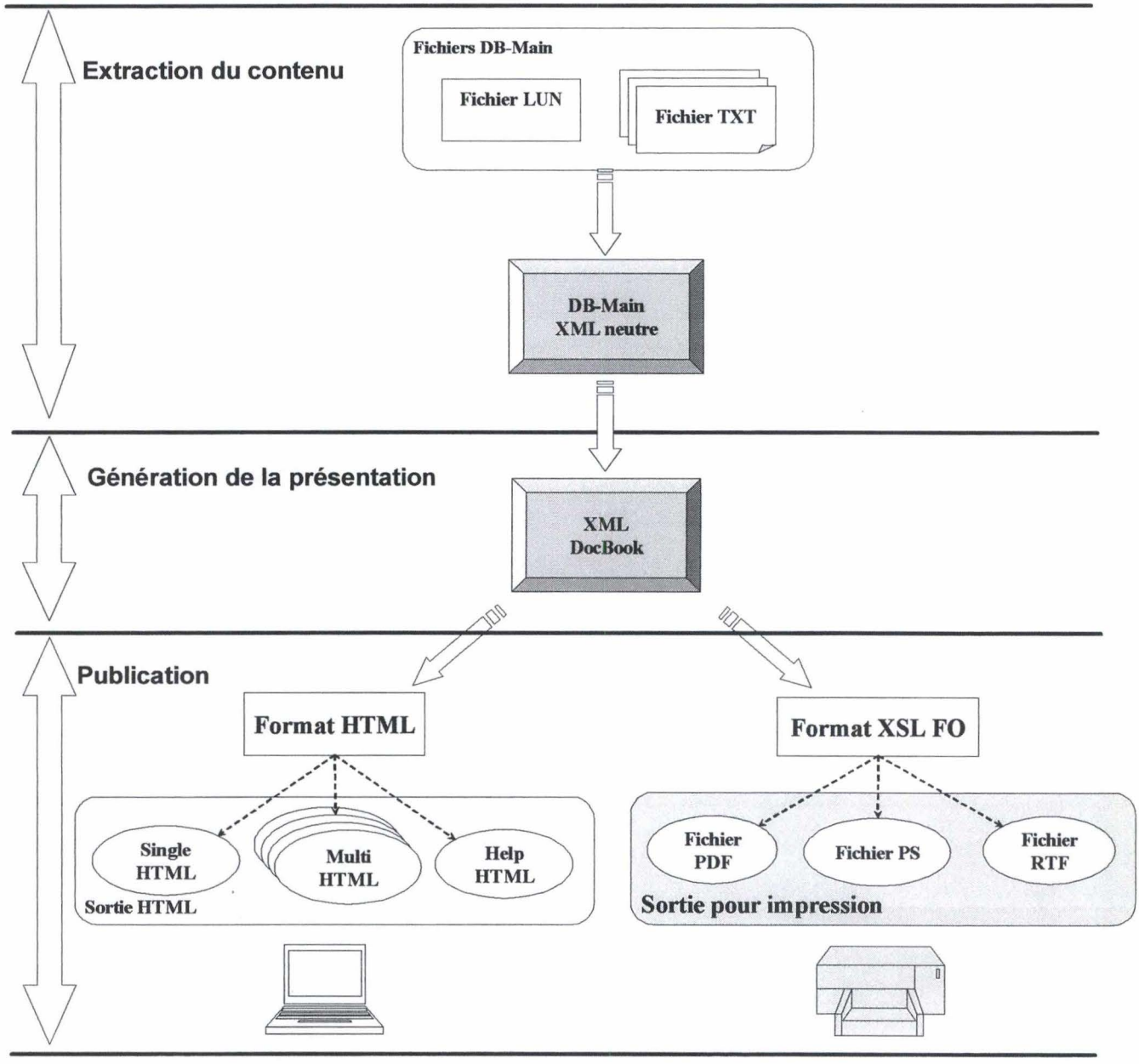

Figure 4 Formats et types de fichiers mis en oeuvre

4.4.2. Principes de séparation

4.4.2.1. Séparation du contenu et de la présentation

La solution d'un plug-in Voyager générant directement un document XML de type DocBook, sans le passage par un document XML intermédiaire (XML neutre), aurait effectivement pu être envisagée, mais cette approche aurait affecté la qualité de la solution en ne respectant pas le principe de séparation du contenu de la présentation, souhaité lorsque l'on met en œuvre les formats de documents de type structurés. La génération d'un document XML neutre avant la création du DocBook permet, par contre, cette séparation claire entre l'extraction de l'information et sa mise en forme.

4.4.2.2. Séparation de la présentation et de la publication

Nous proposons également d'introduire une séparation entre les aspects présentations obtenus avec le document DocBook et la génération des documents physiques. L'objectif est ici de préserver les avantages liés à l'utilisation d'un document XML unique pouvant être converti vers les différents formats de publication.

4.4.3. Modularité et réutilisation de la solution

Ces principes de séparation, que nous suivrons tout au long de l'implémentation du prototype, nous apportent une modularité de la solution qui devrait faciliter la maintenance et l'évolution de l'outil.

Sur le plan de la modularité, la séparation de la partie extraction et de la partie présentation, autorise des modifications importantes du style donné à la présentation, sans remettre en cause le module d'extraction. Les changements de présentation sont en effet obtenus par des modifications au niveau des stylesheets, sans avoir à modifier le programme Voyager.

La séparation de la partie présentation et de la partie publication, autorise pour sa part un développement distinct des différents modules de publication. Les techniques utilisées pour générer les pages HTML sont assez différentes de celles utilisées pour produire les formats d'impression. Les objectifs visés par ces deux types de format le sont également.

Sur le plan de la réutilisation, l'information structurée reprise dans les documents XML neutres générés durant le processus, pourrait avantageusement être utilisée pour d'autres objectifs que la production de documentation.
Ces documents XML pourraient par exemple être réutilisé pour :

- Des analyses ou des traitements sur le contenu des projets. L'information relative aux objets qui est reprise dans ces documents, se trouve sous une forme structurée qui est particulièrement bien adaptée pour les traitements automatiques.

- Des opérations de validation des projets par rapport à des règles qui seraient formalisées sous la forme de DTD.

4.5. Solution pour la documentation de la traçabilité forward/backward

4.5.1. Rappel de l'objectif

L'objectif poursuivi par ce type de documentation est de fournir des indications relatives à l'évolution de l'activité de développement sur l'outil DB-Main. En particulier, on souhaite pouvoir documenter la traçabilité forward/backward qui a pour objectif de tracer les opérations de transformations appliquées aux schémas de manière à pouvoir faire un lien entre les différents composants présents dans ces schémas et ainsi comprendre le processus d'ingénierie appliqué.

4.5.2. Etude de la solution

Documenter la traçabilité forward/backward implique d'exploiter une information qui traite des manipulations appliquées aux objets des schémas. Cette information n'étant pas nativement présente dans le repository des projets, elle doit donc être générée. Pour ce faire, notre première approche fut de tenter d'exploiter la fonctionnalité « Trace » présente dans l'outil et qui permet l'enregistrement des opérations (créations, effacements, transformations, etc.) appliquées aux objets dans un fichier de log.

Malheureusement, cette solution nous est rapidement apparue comme difficile à mettre en œuvre, et ce, pour les raisons suivantes :

- L'information présente dans les fichiers de log est complexe à exploiter, car elle peut contenir toutes les manipulations qu'il est possible d'effectuer sur les objets d'un projet, ce qui représente un très grand nombre de combinaisons possibles. En outre, ces combinaisons d'actions sont appliquées à une grande variété d'objets. Le traitement automatique de cette information est donc particulièrement complexe, car il implique une reconnaissance automatique de toutes les combinaisons possibles.

- La fonctionnalité Trace existante dans DB-Main est limitée à enregistrer des informations sur un seul schéma à la fois. Les relations entre schémas sont donc à reconstruire.

La fonction Trace apparaissant, comme assez complexe à exploiter, nous proposons de mettre en œuvre une autre fonctionnalité de DB-Main qui est la possibilité d'introduire des métas informations sur les objets d'un projet et d'utiliser ces métas propriétés pour générer l'information nécessaire à la documentation de la traçabilité.

La démarche consiste à créer deux métas propriétés additionnelles sur tous les objets que l'on souhaite tracer et à assigner la valeur de l'identifiant technique de l'objet à ces métas propriétés. L'identifiant technique est unique pour chaque objet dans DB-Main. Les métas propriétés étant attachées à l'objet, c'est la conservation de ces métas propriétés au cours des manipulations subies par l'objet qui va nous permettre de conserver la valeur de l'identifiant technique de l'objet d'origine et ainsi de suivre les transformations appliquées.

4.5.3. Principe de fonctionnement

Dans son principe, le fonctionnement de la solution proposée repose sur l'existence de liens entre les objets des schémas créés au cours du processus de développement par transformation de schéma. Pour documenter ce processus, nous allons donc mettre en évidence ces liens dans la documentation générée ce qui nous permettra de suivre les transformations subies par les objets de schéma en schéma.

La mise en œuvre de cette méthode comprend donc deux aspects, avec d'une part, la création et la transmission des métas propriétés entre les schémas, et d'autre part, l'utilisation de ces métas propriétés pour générer des hyperliens dans la documentation des schémas.

4.5.3.1. Création et transmission des métas propriétés.

Pour tous les objets tracés par la méthode, nous définissons d'abord deux nouvelles métas propriétés :

- **CurrentOID** : Méta propriété qui prend la valeur de l'identifiant technique de l'objet courant.
- **ParentOID** : Méta propriété qui prend la valeur de l'identifiant du parent dont est issu l'objet.

Durant le développement, nous appliquons la procédure suivante :

1. Sur le schéma parent de départ, nous appelons un plug-in spécifique : OIDprocess.oxo, qui lors de son exécution va créer, si elles n'existent pas, les 2 métas propriétés supplémentaires et, va assigner la valeur de l'identifiant technique courant de l'objet à la méta propriété CurrentOID et la valeur 0 à la Méta propriété ParentOID.

2. Pour chaque nouveau schéma nécessaire au développement, nous créons le nouveau schéma par copie du schéma parent, puis nous appliquons à ce nouveau schéma les transformations souhaitées. Avec l'opération de copie de schéma, nous avons donc transmis la valeur des métas propriétés CurrentOID et ParentOID des objets au nouveau schéma.

3. Après l'application des transformations, nous utilisons de nouveau le plug-in OIDprocess.oxo sur le schéma transformé, mais cette fois-ci, la valeur de l'identifiant technique de l'objet parent, actuellement assigné à CurrentOID, est transmise à la méta propriété ParentOID du nouvel objet et la valeur correcte de CurrentOID est restaurée avec l'identifiant technique courant de l'objet.

Le résultat de ce traitement est, que nous obtenons sur chaque objet tracé, via la méta propriété ParentOID, l'identifiant technique de l'objet parent et ce malgré les transformations appliquées, car les métas propriétés restent attachées aux objets manipulés.

4.5.3.2. Utilisation des métas propriétés pour documenter la traçabilité

Les informations relatives aux objets parents contenus dans les métas propriétés, vont nous permettre d'enrichir la documentation produite, de liens entre les objets parents et les objets enfants. Au cours de la génération de la documentation des schémas, le plug-in Rep2XML.oxo va extraire ces informations du repository pour les introduire dans le document XML neutre et lors de la construction des pages HTML, le parseur de transformation établira les relations entre les objets en créant des liens dans les 2 sens entre les objets parents et les objets enfants. Nous aurons par exemple, dans le cas d'une transformation d'un schéma conceptuel vers un schéma logique, des liens entre les types d'entité du schéma conceptuel et les tables du schéma logique et inversement.

4.5.4. Avantages et limitations

- Le principal avantage de la méthode proposée est que l'information générée est facile à traiter, puisque ce traitement se limite à une exploration du repository. Il n'y a donc pas de processus de reconnaissance automatique de combinaisons complexes à effectuer pour tracer l'activité.

- La méthode est simple à utiliser et ne demande que peu d'activité additionnelle pour générer l'information nécessaire.

- Cette solution présente également l'avantage de s'intégrer naturellement à l'outil de documentation, car c'est la création d'hyperliens ainsi que le contexte de ces hyperliens dans la documentation qui expliquent le cheminement de l'activité de transformation.

<u>En contrepartie, cette méthode présente certaines limitations :</u>

- Le procédé implique la mise en œuvre de plusieurs schémas, le schéma parent et le schéma transformé, pour suivre et documenter les manipulations appliquées.

- Cette méthode ne donne pas d'indications explicites sur la nature des transformations appliquées, mais nécessite l'interprétation de la documentation fournie (le contexte des liens créés) pour déduire le processus ingénierie suivi.

- L'information de la traçabilité est limitée, car tous les types de manipulations ne peuvent pas être tracés par cette méthode. Par exemple, certaines créations ou certains effacements d'objets effectués ponctuellement dans un schéma ne seront pas tracés. Nous présentons dans la partie implémentation du prototype, les résultats en termes de traçabilité obtenus avec cette méthode lors de l'application du modèle de transformation relationnel.

4.6. Synthèse des justifications de la solution

Les tableaux ci-dessous présentent pour chacun des besoins identifiés, les éléments de la solution qui y répondent.

4.6.1. Réponses aux besoins fonctionnels

Besoins fonctionnels	*Eléments de la solution*
1. Documenter les schémas et les textes	▪ Exploitation des capacités de structuration et d'intégration du format XML ▪ Accès aux informations du repository avec un plug-in Voyager
2. Permettre de documenter les grands schémas.	▪ Robustesse de modèle DocBook. ▪ Métaphore du livre avec DocBook. ▪ Utilisation de liens structurels.
3. Documenter les transformations de schémas.	▪ Utilisation des métas propriétés sur les objets de DB-Main et création de liens hypertextes dans la documentation.
4. Génération automatique de la documentation.	▪ Extraction automatique des informations du repository avec le plug-in Voyager. ▪ Mise en œuvre des transformations XSLT pour générer le DocBook et pour produire les documents.
5. Documentation de type hypertexte.	▪ Génération du format HTML 4.01 par transformation XSLT avec l'utilisation des stylesheets DocBook version 1.61.0.
6. Paramétrage du contenu.	▪ Utilisation de la fonction Mark de DB-Main pour la sélection du contenu.

Figure 5 Justifications pour les besoins fonctionnels

4.6.2. Réponses aux besoins non fonctionnels

Besoins non fonctionnels	*Eléments de la solution*
1. Intégration à DB-Main.	▪ Appel du plug-in Voyager directement depuis l'interface de DB-Main.
2. Solution modulaire et réutilisable.	▪ Principe de séparation contenu/présentation et présentation/publication
3. Utilisation de standards technologiques et de formats normalisés	▪ Utilisation des recommandations W3C XML et HTML. ▪ Utilisation du standard de la documentation technique DocBook.

Figure 6 Justifications pour les besoins non fonctionnels

5. Prototype de l'outil de documentation

Ce chapitre présente le prototype du générateur de documentation que nous avons implémenté. Nous présentons tout d'abord l'architecture de ce prototype, durant cette présentation, nous passons en revue les composants utilisés que nous détaillons. Nous expliquons ensuite le fonctionnement de l'outil et de son interface de paramétrage. Finalement, nous présentons les résultats de nos tests lors de l'utilisation de l'outil avec des exemples de projets DB-Main ainsi qu'une évaluation des résultats obtenus avec la fonction de documentation de la traçabilité forward/backward.

5.1. Architecture et composants du prototype

L'implémentation du générateur de documentation consiste dans ses grandes lignes en un travail d'intégration de composants, avec en particulier l'interfaçage de l'atelier DB-Main avec l'outil documentaire DocBook. Comme indiqué dans la présentation de la solution, nous avons conservé durant toute l'implémentation le principe de séparation contenu/présentation/publication.

Notre architecture reprend ce découpage et nous retrouvons donc 3 parties :

5.1.1. Partie 1 : Contenu

Cette partie a pour objectif la génération du document XML neutre depuis DB-main.

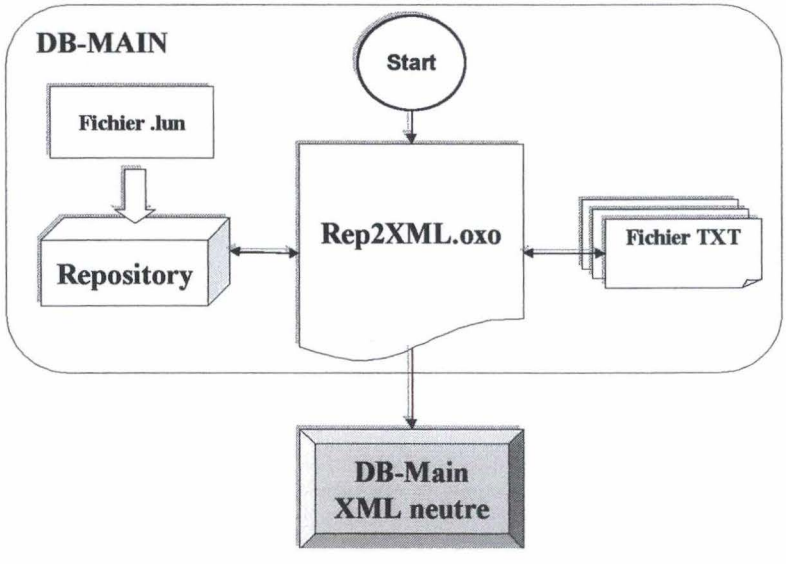

Figure 7 Génération du document XML neutre

Composants :

5.1.1.1. L'atelier DB-Main

De l'atelier DB-Main, nous exploitons le repository et la fonction d'appel des plug-ins de l'interface. Les fichiers impliqués ici sont le fichier .lun du projet et les fichiers textuels associés au projet (fichier .txt, fichier .ddl, fichier .cob, etc.).

5.1.1.2. Le plug-in VOYAGER : Rep2XML.oxo

C'est le principal composant de cette partie de l'architecture, car c'est ce programme Voyager qui démarre le processus de production de la documentation et qui effectue l'extraction des informations du repository. C'est également le composant central de l'outil, car c'est lui qui assure la gestion de l'ensemble des traitements.

Nous présentons dans l'Annexe 1 le détail du fonctionnement du plug-in Rep2XML.oxo.

5.1.2. Partie 2 : Présentation

Dans la partie présentation, le document XML neutre est transformé en document XML DocBook Version 4.2.

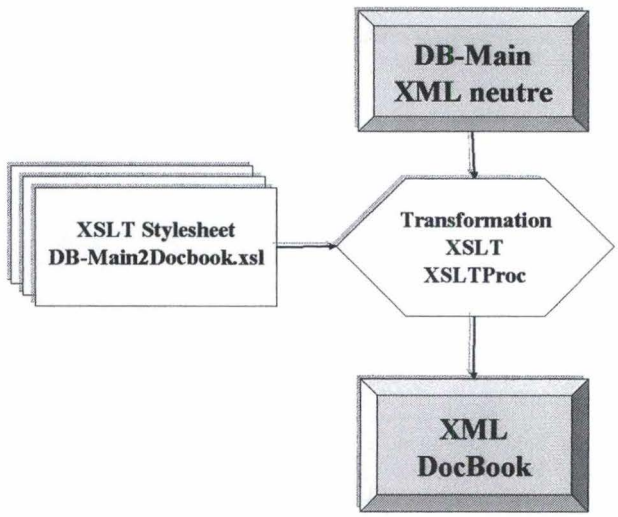

Figure 8 Génération du document XML DocBook

Composants :

5.1.2.1. Le parseur XSLTproc

Issu à l'origine du monde Open Source, le parseur XSLTproc est un parseur rapide écrit en C. XSLTproc fait partie de la librairie de composants XML Libxml2 développée pour la plate-forme UNIX pour le projet Gnome et a été recompilé pour fonctionner sur la plate-forme Win32 (Windows). Ce parseur est distribué sous licence Open source MIT (http://www.opensource.org/licenses/mit-license.html) et peut être utilisé librement.

5.1.2.2. Les feuilles de style DB-Main2DocBook

Ce sont donc les stylesheets développées pour la mise en forme de la documentation de DB-Main. Pour l'élaboration de ces feuilles de style, nous avons également cherché à conserver une approche modulaire pour faciliter le maintien de l'outil, en particulier pour les extensions de documentation envisageables.

Nous retrouvons pour les feuilles de style le découpage suivant :

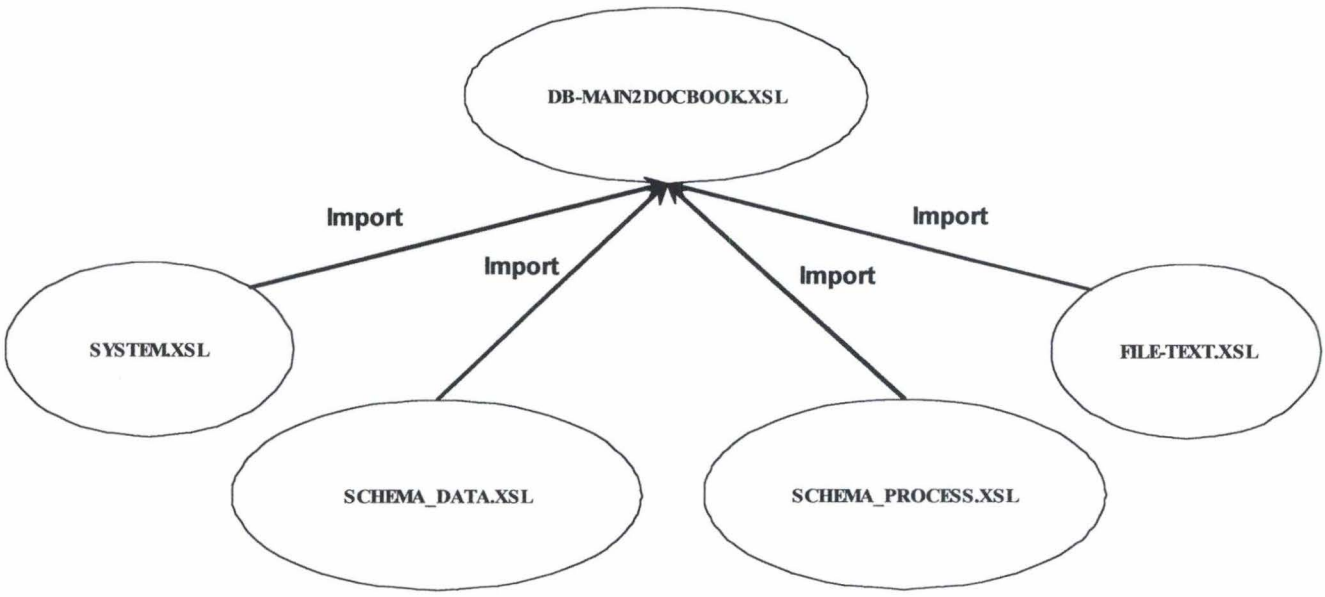

Figure 9 Stylesheets de documentation DB-Main

- DB-Main2DocBook.xsl : Stylesheet principale qui permet la création du DocBook et l'introduction des informations spécifiques à la documentation créée (titre, auteur, date de création, etc.). C'est aussi via cette stylesheet que l'on définit l'allure générale du document avec notamment la position de la table des matières et de l'index.
- System.xsl : Stylesheet qui introduit le chapitre reprenant les informations sur le système du projet et qui comprend : les informations générales sur le projet, les métas propriétés et les domaines définis par les utilisateurs. Nous introduisons également à ce niveau, un tableau reprenant la liste des schémas documentés ainsi qu'un tableau des documents textuels traités.
- Schema-Data.xsl : Stylesheet assurant la mise en forme des informations relatives aux schémas data sélectionnés pour être documentés. Un chapitre est créé pour chaque schéma et chaque chapitre reprend une section pour :
    - Les informations générales relatives au schéma.
    - Les types d'entité
    - Les types d'association
    - Les collections

  Pour chaque section : type d'entité, type d'association ou collection nous générons des sous-sections qui détaillent ces éléments.
- Schema-Process.xsl : Stylesheet pour la mise en forme des schémas process. Ici aussi un chapitre est créé pour chaque schéma. Chaque chapitre de la documentation des schémas process reprend une section pour :
    - Les informations générales du schéma.
    - Les unités process.
    - Les objets données internes.
- File-Text.xsl : Stylesheet destinée à la mise en forme de la documentation des fichiers textes : Texte, SQL, COBOL, etc. Pour chaque document, nous générons un chapitre avec une section proposant des informations générales et une section reprenant le contenu textuel du document.

5.1.3. Partie 3 : Publication

La partie publication comprend tous les éléments nécessaires à la génération des documents finaux.

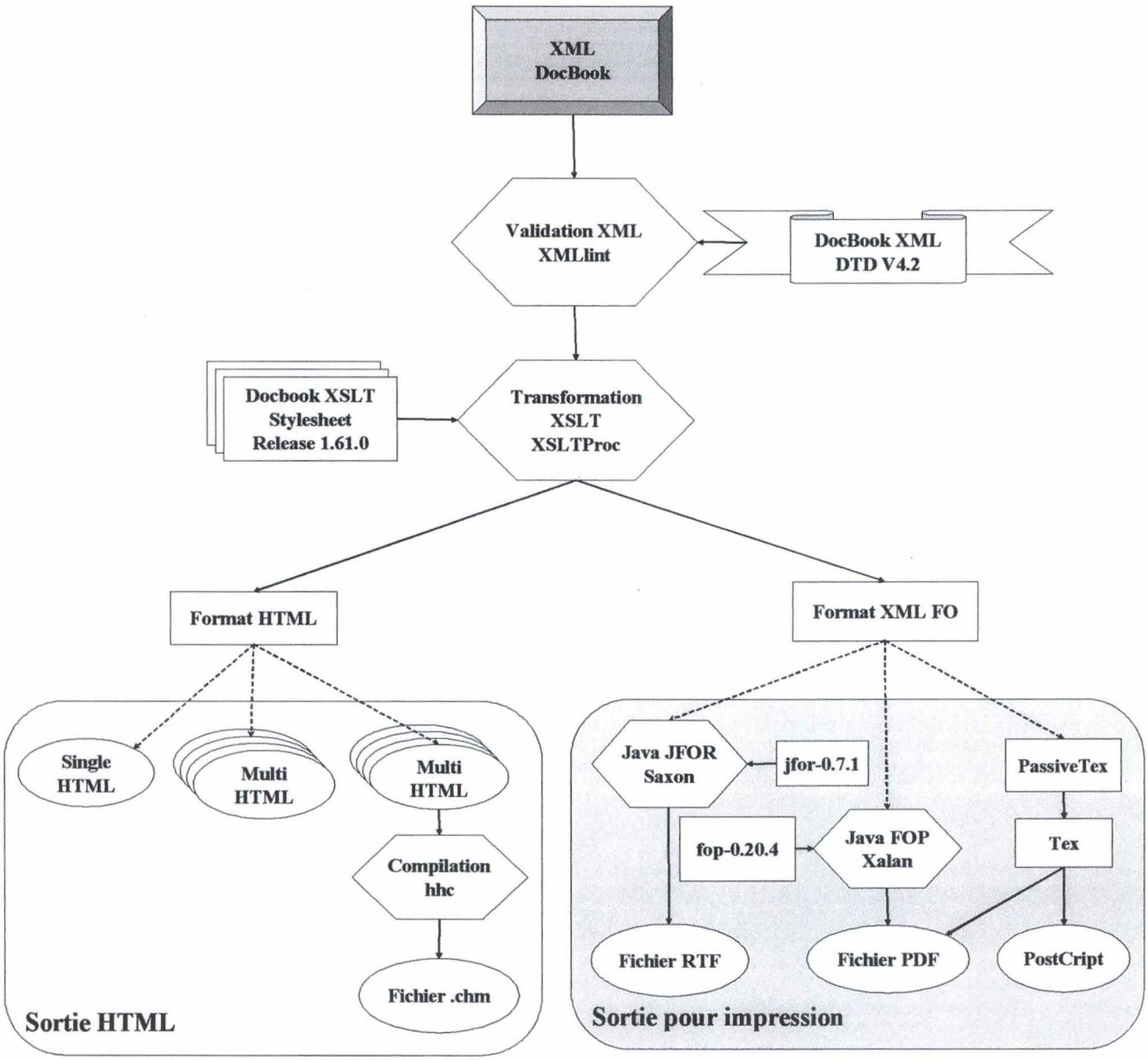

Figure 10 Génération des formats de publication

Composants :

5.1.3.1. Le parseur de validation XMLlint

Le parseur XMLlint fait également partie de la librairie Libxml2. XMLlint est utilisé pour la validation des DocBooks générés.

5.1.3.2. La DTD DocBook V4.2

La DTD DocBook V4.2 propose un vocabulaire structuré très riche (plus de 370 éléments) pour l'écriture de documentations techniques. C'est un atout important de cette solution, car cela nous permet de construire des documents sophistiqués répondant parfaitement à nos besoins. Outre les éléments de structure des documents (chapitres, sections, sous-sections), nous disposons d'un vaste vocabulaire qui permet de décrire des composants complexes tels que les tableaux, les tables des matières, les formats d'images, etc..

5.1.3.3. Les feuilles de style DocBook XSL Release 1.61.0

Ces feuilles de style permettent actuellement la transformation de documents XML DocBook sources vers deux types de publication [Stayton 03] :

1. La publication en ligne sous forme d'hypertexte avec les trois formats HTML possible : Single HTML, Multi HTML et Help HTML.

2. La publication pour l'impression qui est obtenue via le format FO. Les documents FO pour « Formatting Objects » sont des documents XML qui permettent de définir précisément l'apparence physique d'un document destiné à l'impression. Ce format dispose d'un vocabulaire très évolué permettant de formaliser de manière précise la mise en forme des éléments du document (textes, images, graphiques), ainsi que leurs dispositions pour un format papier donné. Il s'agit en fait d'un format intermédiaire qui avec l'aide d'outils de transformation, va nous permettre de produire les formats d'impression courants :
   - PDF : Le format de publication bien connu de l'éditeur Adobe.
   - RTF : Le format accepté par la plupart des traitements de texte.
   - Postscript : Le format pour imprimante.

Sur le plan de la flexibilité et du paramétrage de la mise en forme, les feuilles de style DocBook disposent d'un grand nombre de paramètres qui permettent de contrôler finement le rendu final des documents. Il est de plus toujours possible de modifier directement les feuilles de style XSLT pour adapter l'outil à nos besoins.

5.1.3.4. Les parseurs de transformation XML

Nous retrouvons dans cette partie le parseur XSLTproc déjà utilisé dans la partie présentation et qui va nous permettre ici de convertir le document XML DocBook vers les formats HTML et vers le format FO.
Nous avons aussi deux parseurs Java (Saxon et Xalan), que nous adoptons pour leurs compatibilités avec les outils de transformation vers les formats d'impression PDF et RTF.

5.1.3.5. Les outils de transformation vers les formats d'impression.

Pour produire les formats d'impression, nous utilisons aussi les outils suivants :

- Le processeur FOP (classes Java) pour la transformation du FO en PDF.
- Le processeur XMLmind FO (classes Java), pour la transformation du FO en RTF.
- PassiveTex et Tex pour générer du Postcript (non implémenté).

5.1.3.6. Le compilateur HHC

Le compilateur HHC de Microsoft permet de rassembler dans un seul fichier d'aide de type .chm , les fichiers Help HTML générés par le parseur XSLTproc.

5.2. Fonctionnement du générateur de documentation

Aux 3 parties de l'architecture présentées ci-dessus, correspondent les 3 étapes de la génération de la documentation d'un projet DB-Main. Sur le plan du fonctionnement, l'ensemble du traitement est géré par le plug-in Voyager Rep2xml.oxo qui après l'acquisition des paramètres, pilote successivement les 3 étapes du processus.

5.2.1. 1ére étape : Génération du document XML neutre

L'extraction des informations du repository et la génération du document XML neutre sont effectuées directement depuis le plug-in Rep2xml.oxo. Celui-ci effectue durant son exécution une exploration du projet afin d'en extraire les informations souhaitées et introduit ces informations dans un document XML valide. Nous retrouvons donc dans le document XML obtenu, une structure en arborescence reprenant les objets du projet traité. Nous présentons ci-dessous le schéma de cette arborescence.

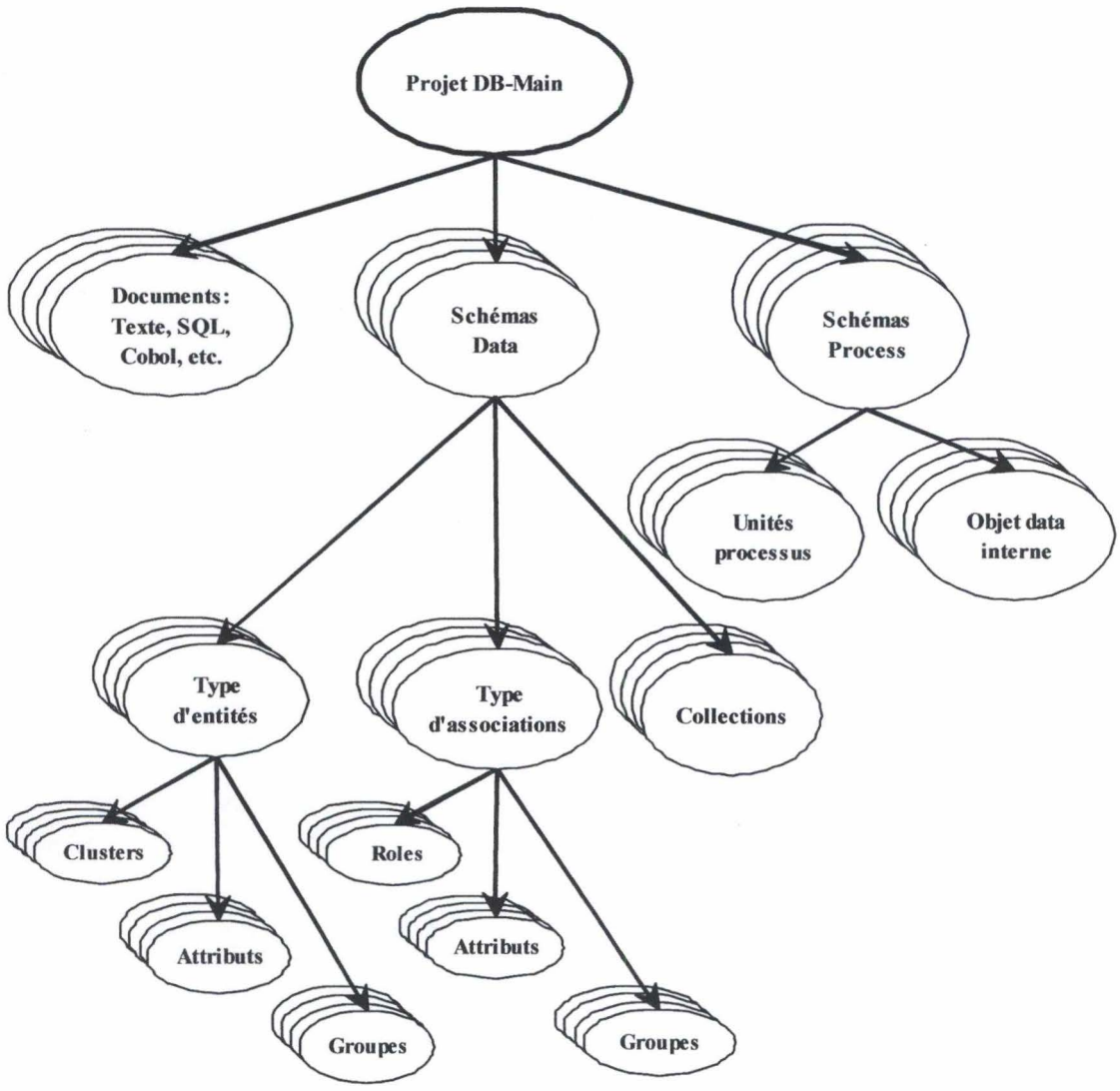

Figure 11 Arborescence du document XML neutre

La DTD pour les documents XML neutre générés par le prototype est présentée en Annexe 2.

5.2.2. 2 ème étape : Génération et validation du DocBook

5.2.2.1. Génération du DocBook

La transformation du document XML neutre en un document DocBook s'obtient donc par des transformations XSLT qui utilisent les feuilles de style développées pour DB-Main. Le traitement est assuré par XSLTproc qui va importer les feuilles de style et effectuer la conversion du document XML neutre vers le document XML DocBook, c'est donc ici une transformation XML vers XML.

5.2.2.2. Validation du DocBook

La validation du DocBook est assurée par le parseur XMLlint qui valide le document par rapport à la DTD DocBook V4.2 et fournit un rapport en cas d'erreur dans le document.

5.2.3. 3ème étape : Génération des formats de publication

La transformation du DocBook vers les formats de publication est également obtenue par des transformations XSLT. Nous avons pour les 2 types de publication :

5.2.3.1. Publication aux formats HTML

Pour les formats Single HTML et multi HTML, la génération des documents s'obtient directement par transformation XSLT via le parseur XSLTproc avec l'utilisation des stylesheets de la distribution standard OASIS DocBook-XSL-1.61.0
Le format Help HTML est lui obtenu en 2 étapes :

- Une première étape qui est la génération d'un ensemble de pages HTML toujours par transformation XSLT, mais avec ici des feuilles de style spécifiques pour les documents Help HTML.
- La seconde étape est un traitement qui collationne l'ensemble des pages Help HTML générées pour les rassembler dans un fichier d'aide de type .chm.
  Ce second traitement est obtenu par l'exécution du compilateur HHC.exe

5.2.3.2. Publication aux formats d'impression

La génération des formats d'impression PDF, RTF et PS est obtenue indirectement avec la mise en œuvre du format FO. Nous avons donc, là encore, un traitement en 2 étapes.

- La génération du format FO par transformation du DocBook, toujours avec des transformations XSLT. Nous utilisons pour ces transformations des stylesheets de la distribution DocBook-XSL-1.61.0 qui sont spécifiques pour ce format et c'est toujours XSLTproc qui assure le traitement.
- La création des documents par la mise en œuvre des parseurs JAVA et des outils qui effectuent la conversion du format FO vers le format de publication cible.
  - Pour la production des documents PDF, nous utilisons les classes Java fop-0.20.4 qui mettent en œuvre le parseur XALAN.
  - Pour la production des documents RTF, nous utilisons le processeur XMLmind FO qui met en œuvre le parseur SAXON.
  - Pour la génération des fichiers PS, nous utilisons le convertisseur PassiveTex.

Remarque :
La production des formats d'impression n'a pas été développée dans le cadre de ce travail. Pour les formats PDF et RTF, nous nous sommes limités à tester le bon fonctionnement de la génération des documents avec les outils Java.

5.3. Paramétrage

L'interface de paramétrage de l'outil conserve également dans sa présentation, une séparation claire entre les paramètres relatifs au contenu, les paramètres relatifs à la présentation et les paramètres relatifs aux formats de publication des documents.

5.3.1. Paramétrage du contenu

Le paramétrage du contenu consiste en la sélection des éléments du projet qui seront documentés. Cette sélection s'opère sur 2 modes avec d'une part une sélection sur les groupes d'objets à documenter et d'autre part une sélection sur les types d'objets à documenter.

1. Sélection des groupes d'objets
   Les objets sont ici sélectionnés au préalable, avec la fonction Mark de DB-Main et une option du menu content de l'interface de paramétrage permet de limiter la documentation aux objets marqués. Cette sélection par marquage comporte elle-même deux niveaux, avec d'abord une sélection au niveau des schémas et des documents à traiter, puis une seconde sélection au niveau des objets à documenter dans chaque schéma.

2. Sélection du type d'objet
   L'interface propose également de spécifier le type d'objets à documenter.
   Avec le choix de documenter au premier niveau les types de production : schémas data, schémas process, documents et au second niveau, les types d'objets : types d'entité, types d'association, collections, groupes, attributs, etc.

C'est aussi lors du paramétrage du contenu que la documentation de la traçabilité forward/backward peut être demandée. Par la sélection d'une option du menu contenu, les traitements relatifs à la traçabilité sont effectués et les liens entre les objets sont introduits dans la documentation générée.

5.3.2. Paramétrage de la présentation

Cette partie permet de définir la mise en forme de la documentation générée.
Ces paramètres pouvant être, soit communs à l'ensemble des formats de publication, soit spécifiques à certains, nous proposons d'une part, des paramètres généraux communs, comme par exemple la présence d'une table des matières, la création d'un index, etc., et d'autre part, un paramétrage spécifique pour chaque type de format. Nous avons, par exemple, la sélection du niveau de découpage du document en page HTML pour le format Multi HTML et Help HTML.

5.3.3. Paramétrage du format de publication

Ce dernier paramètre permet simplement de choisir le format de sortie souhaité parmi les possibilités suivantes : Format XML, Single HTML, Multi HTML, Help HTML, PDF et RTF.

5.3.4. Implémentation de l'interface de paramétrage

L'interface proposée est implémenté en Visual Basic et est appelée par le plug-in Rep2XML.oxo dès le démarrage du traitement.

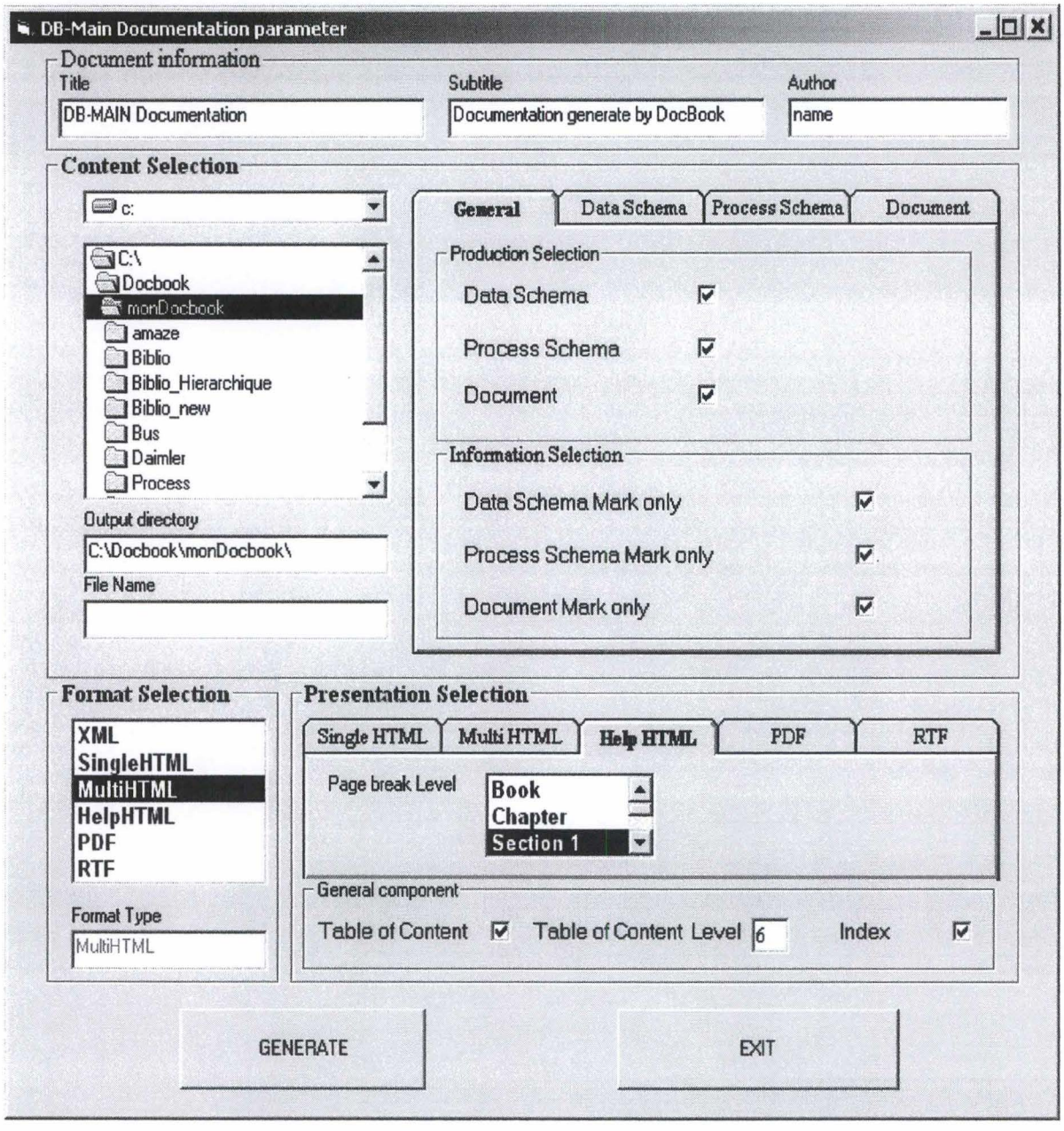

Figure 12 Interface de paramétrage du générateur

L'installation et l'utilisation du prototype de l'outil et de son interface sont présentées en détail dans l'Annexe 3.

5.3.5. Fonctionnement

Avec le démarrage du plug-in Rep2xml.oxo depuis DB-Main, l'interface de paramétrage de l'outil est affichée pour permettre à l'utilisateur d'introduire ses choix. Une fois les paramètres souhaités sélectionnés et validés, le programme génère deux fichiers de paramétrage qui seront exploités durant l'exécution automatique de l'outil.

- Le premier fichier obtenu est un simple fichier texte : param.txt qui reprend le paramétrage du contenu de la documentation. Ce fichier est utilisé par Rep2xml.oxo pour sélectionner les composants à documenter lors de l'extraction des informations du repository.
- Le second fichier est en fait une stylesheet qui reprend le paramétrage de la présentation et qui sera exploitée lors des différentes transformations XSLT.

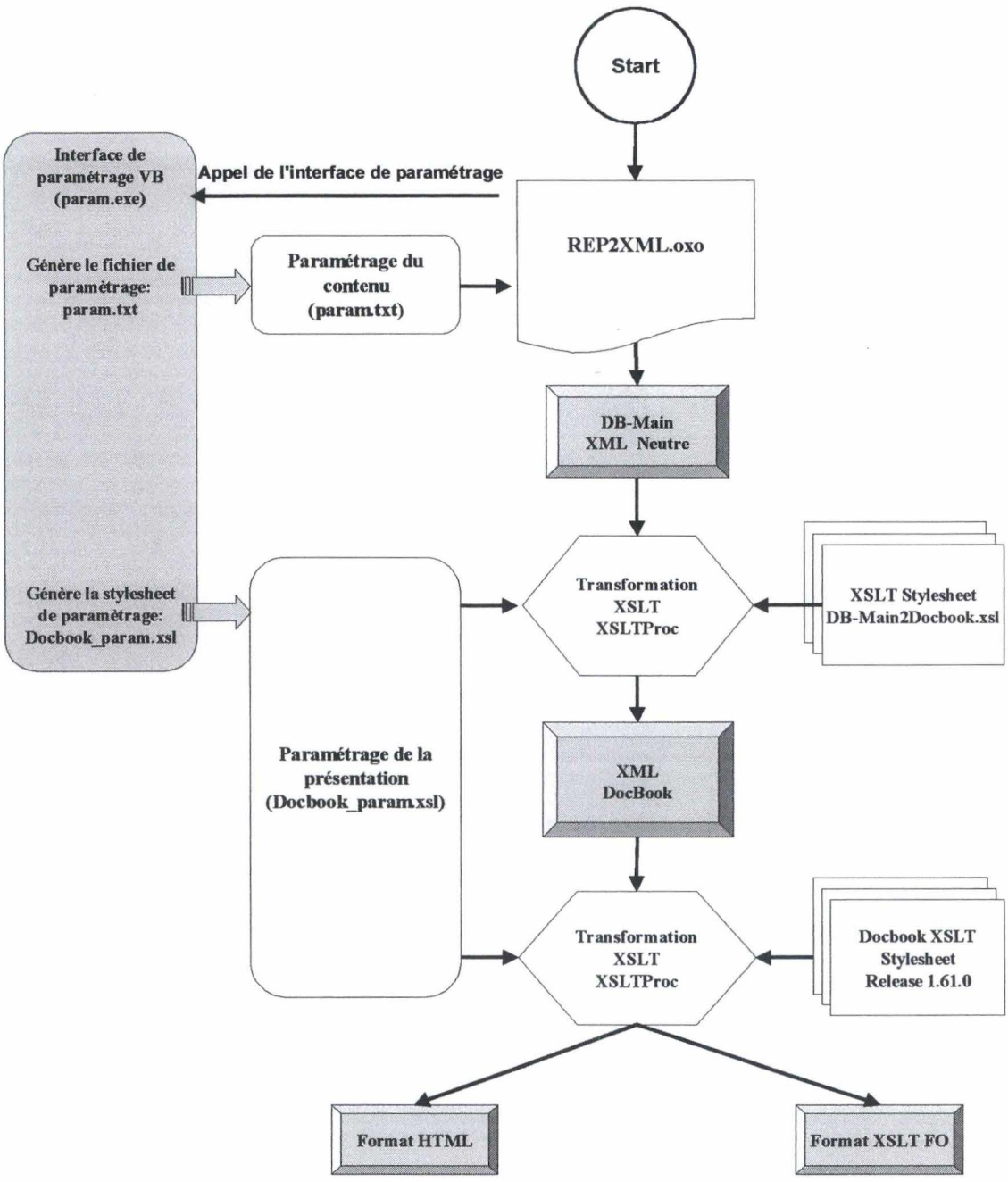

Figure 13 Fonctionnement du paramétrage du générateur

5.4. Evaluation du générateur de documentation

Pour évaluer notre prototype, nous avons effectué une série de tests en soumettant à l'outil plusieurs projets DB-Main. Nous présentons dans le tableau ci-dessous le détail des résultats obtenus. Pour chaque projet, nous présentons le nombre de schémas et de documents traités et le nombre de pages HTML générées, nous indiquons également le temps de traitement nécessaire et la taille du DocBook obtenue.

Nom du projet	*Nombre de schémas*	*Nombre de documents*	*Nombre de pages HTML*	*Temps de traitement (min)*	*Taille du DocBook (Mb)*
Academique	1	0	53	1	0.40
Biblio	4	2	80	2	0.62
Bus	3	0	83	2	0.57
Bib-Hierachique	7	2	137	2	0.88
Purchasing	9	84	632	23	3.34
schéma test	**3**	**0**	**876**	**198**	**20.36**

Figure 14 Tests d'évaluation de la génération de documentation

Pour l'ensemble de ces projets, la documentation a été générée au format Help HTML et Multi HTML sans difficultés. Sur le plan de l'archivage, le format Help HTML apparaît beaucoup plus commode à manipuler, car il rassemble dans un seul fichier comprimé les nombreuses pages HTML générées.

Nous présentons en Annexe 4, la documentation au format RTF générée pour la partie schéma conceptuel du projet BIBLIO. Cette annexe montre également des extraits de la documentation produite au format Help HTML.

Documentation des grands schémas

Pour évaluer la génération de la documentation de grand schéma, nous avons soumis à l'outil un projet DB-Main relatif à une base de données réelle (schéma test) comptant plus de 800 objets (types d'entité, types d'association, tables) répartis sur plusieurs schémas.

Le résultat obtenu est la création d'un document de près de 900 pages HTML. Ce test confirme la robustesse de l'outil de documentation DocBook pour ce type de traitement.

Par contre, le temps de traitement pour générer cette documentation est assez long. En effet au niveau des transformations XSLT vers les formats de publications, malgré l'utilisation du parseur rapide XSLTProc, nos essais ont mis en évidence un temps d'exécution plus important pour le traitement des grands schémas. Pour l'exemple de documentation dont il est question ci-dessus, alors que les transformations ne prennent que quelques minutes pour générer le document XML neutre et le DocBook, la production des pages HTML a demandé plus de trois heures d'activité de la machine.

Bien que ces délais de traitements reste raisonnables, au regard du volume de documents générés, des améliorations pourraient être apportées par une optimisation du processus notamment dans au niveau de l'utilisation de la mémoire lors du chargement des stylesheets utilisées pour générer les pages HTML.

5.5. Implémentation et évaluation de la documentation de la traçabilité

5.5.1. Implémentation

Pour implémenter cette partie de l'outil, nous faisons appel à plusieurs composants :

- L'élément principal est ici le plug-in OIDprocess.oxo qui assure la création des métas propriétés CurentOID et ParentOID ainsi que l'assignation de la valeur des identifiants techniques à ces métas propriétés. Les métas propriétés CurentOID et ParentOID sont créées pour les objets suivants :

  - Les types d'entité
  - Les types d'association
  - Les collections
  - Les attributs
  - Les groupes
  - Les rôles

- Le plug-in Rep2XML.oxo est également impliqué, puisque, outre son rôle dans la génération de la documentation, c'est lui qui effectue l'exploration préalable des métas propriétés du projet lorsque la documentation de traçabilité est demandée. Cette opération est nécessaire pour collecter les données qui permettent d'assurer la traçabilité dans les deux sens comme souhaité. Le traitement effectué par le plug-in Rep2XML.oxo pour la documentation de la traçabilité est le suivant :

  - Au démarrage du traitement, le programme exécute une recherche sur les objets du projet afin de collecter les informations relatives à la traçabilité et constitue durant cette recherche une liste reprenant pour chaque objet, la valeur des métas propriétés CurrentOID et ParentOID ainsi que l'identifiant de tous les objets ayant l'objet traité comme parent. Ceci nous permet d'obtenir pour chaque objet la liste de ses descendants.

  - Lors de la phase d'extraction des informations de documentation du projet, pour chaque objet documenté, le programme recherche dans la liste créée les objets enfants de l'objet traité ce qui permet d'établir la relation parent/enfant qui correspond à la traçabilité forward. La traçabilité backward est elle obtenue directement, via la méta propriété ParentOID qui défini la relation enfant/parent pour l'objet que l'on document.

  - Ces informations sur la traçabilité sont ensuite introduites dans le document XML neutre au niveau de la documentation de l'objet. Nous utilisons pour cette partie de la documentation un balisage spécifique dont voici un exemple :

```
<trace>
<parent_OID type="Relation-type" parent_Name="emprunte"parent_OID="id54"/>
<shild_OID type="Entity-type" shild_Name="emprunte" shild_OID="id3615"/>
</trace>
```

- Le parseur de transformation XSLTproc est aussi mis à contribution

  - D'abord, durant la création du Docbook, il met en forme les informations contenues au niveau des balises relatives à la traçabilité en les plaçant dans des tableaux dans une sous-section de la documentation de l'objet.
  - Ensuite, lors de la production des documents en introduisant les liens hypertextes qui relient les objets entre eux dans les pages HTML.

5.5.2. Intégration à l'outil de documentation

L'intégration à l'outil de documentation est assez naturelle puisqu'il s'agit en fait d'introduire des liens supplémentaires dans la documentation produite. Au niveau de l'interface de paramétrage, nous avons simplement ajouté un sélecteur dédié dans la partie sélection du contenu qui permet d'activer la documentation de la traçabilité.

5.5.3. Utilisation de la documentation de la traçabilité forward/backward

Dans le cadre du développement d'une base de donnée par transformations de schéma, l'utilisation de la méthode de documentation proposée implique d'abord la création et la transmission des métas propriétés CurrentOID et ParentOID sur les objets tracés durant les étapes de conception. Pour le projet BIBLIO présenté ci-dessous, nous avons donc la création des métas propriétés au niveau du schéma conceptuel et la transmission de ces métas propriétés au niveau du schéma logique et du schéma physique durant le développement. Le traitement est obtenu avec l'utilisation du plug-in OIDprocess.oxo.

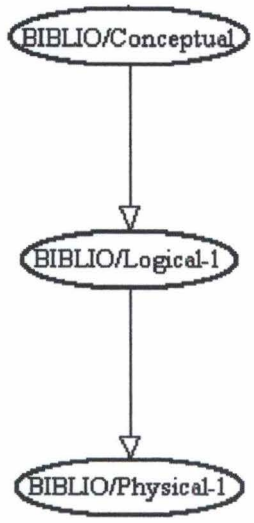

Figure 15 Etapes de conception du projet BIBLIO

Ensuite, la documentation sur la traçabilité est générée automatiquement lors de la production normale des documents.

Pour l'interprétation de la documentation générée, c'est le contexte des hyperliens présents dans la documentation produite qui permet de comprendre les transformations appliquées aux objets durant le développement.

Pour illustrer notre propos, nous présentons ci-dessous un extrait de la documentation du schéma logique du projet Biblio (BIBLIO/Logical-1), qui montre la documentation de la traçabilité obtenue sur l'objet emprunte dans la section type d'entité. Nous présentons ensuite l'interprétation de ces informations.

2.3. emprunte

2.3.1. GENERAL INFORMATION

- Name: emprunte
- Short name: CLO
- Technical Identification: id3318

2.3.2. DESCRIPTIONS

Type	Description
Semantic	History of a former borrowing. When a copy is brought back, the information on its borrowing is kept, making up a so-called "closed borrowing". There cannot exist more than one closed borrowing for a given copy at the same date. No

2.3.3. TRACE FORWARD BACKWARD

Backward

Type	Name
Relation-type	emprunte

Forward

Type	Name
Entity-type	emprunte

Figure 16 Documentation de la traçabilité de l'objet emprunte

Au point 2.3.3 TRACE FORWARD BACKWARD de la documentation de l'objet emprunte, nous trouvons deux tableaux, le premier tableau présente la traçabilité backward avec un lien exprimant la relation enfant/parent et le second tableau présente la traçabilité forward avec un lien pour la relation parent/enfant.
Ces indications au niveau du schéma logique de la documentation, nous permettent de déduire le type de transformation appliqué à l'objet. En effet, la documentation backward nous indique que l'objet d'origine est de type « relation » ce qui correspond donc à la transformation d'un type d'association du schéma conceptuel, en un type d'entité dans le schéma logique ou plus précisément, d'un type d'association en une table. Le lien hypertexte vers le type d'association emprunte du schéma conceptuel matérialise cette transformation.
Pour la traçabilité forward, le lien vers le type d'entité emprunte du schéma physique nous montre simplement que l'objet n'a pas été transformé durant cette partie de la conception.

5.5.4. Evaluation de la traçabilité obtenue avec le plan de transformation : Modèle relationnel

Comme nous le mentionnons lors de la présentation de la solution, la méthode proposée ici offre l'avantage de la simplicité et d'une bonne intégration à l'outil de documentation, mais cette méthode présente aussi certaines limitations en ce qui concerne les opérations sur les schémas qui peuvent être tracées. De manière à évaluer ces limitations, nous présentons ici le résultat obtenu lors de la documentation de la transformation du schéma conceptuel du projet biblio.lun vers un schéma logique avec le plan de transformation : Modèle relationnel.

Nous empruntons à l'ouvrage *« Ingénierie des bases de données »* [Hainaut 00], le plan de transformation suivant :

Etapes	*Opérations appliquées*
1.	Transformation des relations IS-A
2.	Transformation des types d'association complexes
3.	Transformation des attributs composés monovalués de niveau 1
4.	Transformation des attributs multivalués de niveau 1
5.	Répétition des étapes 3 et 4 (si nécessaire)
6.	Transformation des types d'association un-à-un ou un-à-plusieurs
7.	Ajout des identifiants techniques et répétition de l'étape 6 (si nécessaire)
8.	Transformation des identifiants facultatifs
9.	Finalisation

Figure 17 Plan de transformation : Modèle relationnel

5.5.4.1. Résultats obtenus

Pour évaluer la traçabilité obtenue, nous vérifions pour chaque étape du plan de transformation si, au niveau des objets transformés, la méta propriété ParentOID obtenue permet bien de retrouver l'objet parent et donc d'établir le lien avec celui-ci.

Si nous reprenons étape par étape ce plan de transformation, nous obtenons pour la traçabilité les résultats suivants :

Etape 1. Transformation des relations IS-A en type d'association un-à-un
Résultat : Oui, mais indirectement
Commentaire : L'information est obtenue indirectement via la méta propriété ParentOID de l'attribut identifiant des sous-types. La documentation de la traçabilité des relations IS-A transformées implique donc un traitement additionnel pour ramener l'information au niveau des types d'association.

Etape 2. Transformation des types d'association complexes et plusieurs-à-plusieurs en types d'entité avec des types d'association un-à-plusieurs
Résultat : Oui
Commentaire : La traçabilité est obtenue directement et permet la création dans la documentation, d'un lien pointant de l'entité créée vers la relation transformée et inversement, pour la traçabilité forward, de la relation transformée vers l'entité créée.

Etape 3. Transformation des attributs composés monovalués de niveau 1 par désagrégation.
Résultat : Oui, mais indirectement
Commentaire : Nous obtenons bien pour les attributs qui composent l'attribut composé, les liens vers les nouveaux attributs simples constitués. Mais l'attribut composé lui disparaît dans la transformation et il n'y a donc pas de méta propriété ParentOID directement exploitable pour créer un lien. L'introduction d'une documentation de traçabilité au niveau de cet attribut composé demande également un traitement supplémentaire.

Etape 4. Transformation des attributs multivalués de niveau 1 en types d'entité.
Résultat : Oui, mais indirectement
Commentaire : L'information de traçabilité se retrouve ici au niveau de l'attribut identifiant du type d'entité créé et demande donc aussi un traitement pour être présentée directement au niveau du type d'entité.

Etape 5. Répétition des étapes 3 et 4 (si nécessaire)
Résultat : Nous obtenons un résultat identique.

Etape 6. Transformation des types d'associations un-à-un ou un-à-plusieurs en clés étrangères.
Résultat : Oui partiellement.
Commentaire : Pour les types d'associations existants dans le schéma de départ, la traçabilité est obtenue directement avec un lien entre le type d'association et la clef étrangère créée et inversement de la clef étrangère vers le type d'association. Pour les types d'associations un-à-plusieurs créés au cours des transformations, comme c'est le cas des types d'associations créés lors de l'étape 2, la traçabilité n'est pas conservée.

Etape 7. Ajout des identifiants techniques et répétition de l'étape 6 (si nécessaire).
Résultat : Non
Commentaire : Il n'y pas de conservation de la traçabilité par la transmission de méta propriété lors des actions de création et d'effacement d'objet. Par contre, la création d'un objet dans un schéma introduit une valeur 0 pour la Méta propriété Parent_OID de cet objet. Cet indice pourrait être exploité comme indicateur de création d'objet dans un traitement.

Etape 8. Transformation des identifiants facultatifs en types d'entité
Résultat : Sans objet
Commentaire : Cette étape n'est actuellement pas traitée par le script de transformation globale : Relationnal modèle de l'outil.

Etape 9. Finalisation.Transformation des noms pour compatibilité avec la syntaxe SQL.
Résultat : Non.
Commentaire : La documentation obtenue avec l'outil de documentation sans la fonction trace inclut déjà la documentation relative au nom des objets. Les modifications des noms seront donc documentées via les liens déjà présents.

Les résultats de la transmission de la méta propriété ParentOID entre les objets sont résumés dans le tableau suivant :

Etapes	*Objet du schéma de départ*	*Objet du schéma transformé*
1.	Attribut identifiant du super-type	Attribut identifiant du sous-type
2.	Type d'association complexe ou many-to-many	Type d'entité obtenu
3.	Attribut composé Attribut composant	Non transmise, car l'attribut composé n'existe plus Attribut simple
4.	Attribut multivalué	Attribut identifiant du type d'entité obtenu
5.	Idem	Idem
6.	Types d'association du schéma de départ Types d'association créés durant les transformations	Groupe clef étrangère Non transmise ParentOID =0
7.	Sans	Pas de transmission lors de création d'objet
8.	Identifiant facultatif	Non traité
9.	n/a	n/a

Figure 18 Transmission de la méta propriété ParentOID entre objets

5.5.4.2. Evaluation

Il apparaît donc 3 cas possible :

1. La traçabilité est conservée et se retrouve directement sur la documentation des objets. La documentation relative à cette traçabilité est présente dans les documents générés par l'outil pour : les types d'entité, les types d'association, les attributs et les groupes
2. La traçabilité est conservée, mais de manière indirecte et demande une interprétation ou un traitement complémentaire pour être exploitée. Il est à noter que ces traitements peuvent être assez complexes pour obtenir un résultat parfaitement satisfaisant.
3. La traçabilité est perdue au cours des transformations. Il s'agit principalement des cas où il y a création ou effacement des objets. Nous pouvons toutefois envisager, pour certain cas, la possibilité de déduire la création des objets par un traitement qui exploiterait la présence de Méta propriété ParentOID avec une valeur à 0.

Il ressort de ces éléments que l'utilisation des métas propriétés pour documenter la traçabilité forward/backward est une solution intéressante, mais incomplète. Cette approche nécessite à l'évidence le développement de traitements complémentaires pour exploiter davantage l'information disponible.

En l'état, cette solution apporte au lecteur un complément d'information, qui lui permet certes de suivre les principales transformations appliquées de schéma en schéma, mais qui demande un effort d'interprétation important de sa part. L'intérêt pratique de cette documentation devra être jugé à l'utilisation.

6. *Conclusions*

Ce mémoire nous a tout d'abord permis de mettre en évidence l'importance de la documentation dans le processus logiciel. Pour les développeurs de base de données comme pour les autres développeurs, la documentation constitue un outil de travail essentiel au même titre que les environnements de développement évolués proposés aujourd'hui. Pourtant, les études menées sur ce sujet sont unanimes pour souligner les lacunes dans ce domaine. Il existe donc bien un véritable besoin de produire cette documentation. Ce besoin est d'ailleurs démontré par le développement de l'activité de rétro-ingénierie.

Le second élément mis en évidence est que la qualité d'une documentation doit être jugée sur son utilisation effective. Nous en concluons que la conception d'un outil documentaire doit être guidée par les besoins rencontrés et en particulier que les documents produits doivent parfaitement répondre aux attentes des lecteurs.

Sur le plan des technologies existantes, l'état de l'art de l'informatique documentaire effectué, nous apprend que si en matière de systèmes de documentation les solutions sont très variées, nous retrouvons des tendances communes qui sont d'une part, l'emploi de formats de documents structurés, principalement le format XML et d'autre part, l'utilisation du format HTML comme support de publication.

En effet, le format de documents structurés XML constitue actuellement une solution particulièrement attractive pour la génération et la publication de documentation. Pour le cas d'une documentation relative aux bases de données, nous avons eu confirmation de l'intérêt de ce format pour la représentation des schémas ainsi que pour la prise en compte de différentes sources d'information. Les capacités de structuration et d'intégration constituent un des points forts de XML.

L'intérêt de l'hypertexte pour la consultation de la documentation a également été confirmé. Dans notre contexte, avec les tailles importantes des documents produits, le support de la navigation hypertextuelle s'avère indispensable. La rapidité et la facilité de consultation sont ici des facteurs de succès déterminants, en particulier pour l'acceptation des documents produits par les lecteurs. En outre, en adoptant le format HTML comme support de publication, nous bénéficions des avantages liés aux navigateurs standards qui sont déjà présents sur les postes de travail et dont l'utilisation est bien connue du public visé par l'outil.

Les avantages d'une documentation de type hypertexte étant posés, il nous a fallu tenir compte du risque de désorientation engendré par la consultation de documents volumineux. En effet, ce problème qui est bien connu est particulièrement présent lors de la consultation de documents électroniques. En réponse à ce problème, nous avons proposé d'inclure dans les documents de nombreux liens de type structurels qui apportent une organisation aux documents et qui facilitent grandement sa consultation.

L'outil de documentation DocBook a pour sa part bien présenté les qualités attendues de robustesse et d'ouverture, ceci est à remarquer, car ces qualités sont habituellement contradictoires. D'une part, nous avons pu démontrer la robustesse de l'outil lors de nos tests. Nous avons notamment pu produire la documentation complète d'un projet DB-Main comptant plus de 800 objets (types d'entité, types d'association, tables) répartis sur 3 schémas différents et conduisant à la création d'un document de près de 900 pages HTML, au prix il est vrai d'un certain temps de traitement de la part de la machine. D'autre part, la flexibilité et l'ouverture de l'outil nous ont permis de construire des documents répondant parfaitement à nos besoins. Nous avons bien sûr pu utiliser les nombreux paramètres fournis avec l'outil pour adapter la mise en forme des documents à nos souhaits, mais la souplesse de mise en œuvre de l'outil DocBook se manifeste aussi par l'ouverture complète du code source qui nous a permis par exemple, d'inclure nos propres feuilles de style dans le processus de traitement.

Nous pouvons aussi souligner, un autre apport de DocBook qui nous paraît particulièrement intéressant et qui est la métaphore du livre proposée par ce modèle. L'utilisation d'une métaphore familière pour le lecteur constitue certainement une aide appréciable pour faire face aux problèmes d'ergonomies rencontrés lors de la consultation de grands documents hypertextes.

Pour l'implémentation de la solution, nous nous sommes attachés à respecter les principes de séparation proposés entre les parties extraction et présentation et entre les parties présentation et publication. Cette démarche présente l'avantage d'introduire une bonne modularité entre les composants de l'outil ce qui s'est avéré fort utile étant donné les différentes techniques mises en œuvre. Il faut remarquer que cette modularité est facilitée par l'utilisation du format XML qui joue ici son rôle de format générique compatible avec les différents composants du dispositif. Cet aspect est particulièrement bien illustré par la chaîne de production utilisée pour les formats d'impression, où trois types de document XML successifs sont utilisés pour relier l'extraction de l'information depuis le plug-in Voyager, à la production des documents PDF ou RTF par les outils JAVA.

En terme de perspectives, deux évolutions permettraient de compléter l'outil. La première évolution intéressante de ce travail, pourrait être le développement d'une application de gestion de la documentation produite. Cette application serait basée sur un SGBD de type XML qui exploiterait les documents XML DocBook générés. L'intérêt serait double. Tout d'abord, la mise en place de cette application fournirait une solution pour la gestion et l'archivage des documents, une réponse à ces problèmes sera sans doute nécessaire si l'outil doit être utilisé pour documenter de grands systèmes. En suite, une base de données de type XML devrait permettre d'exploiter pleinement le potentiel de l'information contenue dans les documents XML produits, en offrant par exemple, la possibilité d'introduire des requêtes complexes pour retrouver l'information. La seconde évolution possible de l'outil, serait de mettre en place un service de publication en ligne de la documentation dans le cadre d'une infrastructure de type Web. L'objectif serait ici d'améliorer l'accès aux documents, la disponibilité et l'accessibilité sont également des qualités importantes d'une bonne documentation. Avec ces deux extensions, nous disposerions d'un outil documentaire complet, intégrant la génération, la gestion et la distribution de la documentation produite.

Enfin, sur un plan personnel, ce travail constitue pour moi une expérience particulièrement intéressante et riche d'enseignements notamment pour la variété des activités abordées. J'ai tout d'abord été confronté aux aspects théoriques de la recherche et de l'étude bibliographique qui m'ont permis de faire le point sur la problématique de la documentation et sur l'état de l'art dans ce domaine. Cette partie fut suivie d'une phase de réflexion et de synthèse des problèmes et des solutions actuellement disponibles, pour aboutir au choix de la solution défendue ici. Ensuite, la réalisation du prototype m'a permis de mettre en œuvre les solutions étudiées et d'approfondir mes connaissances notamment en découvrant des technologies nouvelles pour moi. J'ai notamment pu me familiariser avec l'outil de développement DB-Main et son langage de programmation Voyager, dont j'ai apprécié les qualités d'évolution et d'ouverture. Ce travail m'a également permis d'aborder les techniques relatives aux traitements du format XML et plus particulièrement les techniques de transformations XSLT. Ces technologies sont, à n'en pas douter, appelées à jouer un rôle de premier plan dans les systèmes d'information de demain.

7. Glossaire

Voici le résumé des acronymes utilisés dans le mémoire :

DB	DataBase
DTD	Document Type Definition
DSSSL	Document Style Semantics and Specification Language
DITA	Darwin Information Typing Architecture
ERP	Enterprise Resource Planning
FO	Formatting Objects
HTML	Hypertext Markup Language
HyTime	l'Hypermedia Time-based Document structuring Language
LIDB	Laboratoire d'Ingénierie des Bases de Données
ODA	l'Office Document Architecture
PDF	Portable Document Format
PS	Post Script
RTF	Rich Text Format
SGBD	Système de Gestion de Base de Données
SGML	Standard Generalized Markup Language
SI	Système d'information
TEI	Text Encoding Initiative
W3C	World Wide Web Consortium
XML	eXtented Markup Language
XSLT	eXtensible Stylesheet Language

8. Bibliographie

Références bibliographiques utilisées pour ce mémoire. Nous avons fourni les liens URL pour les documents disponibles en ligne.

[Ben Romdhane 01]
Ben Romdhane M. : *Navigation dans un espace textuel, accès à l'information scientifique* Thèse de doctorat en Sciences de l'Information et de la Communication, 2001

[Chapin 85]
Chapin, N. : "Software Maintenance: A Different View", AFIPS Conference Proceeding, 54th National Computer Conference, 1985.

[Conklin 87]
Conklin J. : *Hypertext : an introduction and survey*. In « IEEE computer », 1987.

[Englebert 02]
Englebert V. : *Voyager 2 reference manual*, Version 6.5,The University of Namur - LIBD, 2002

[Forward 02 a]
Forward A., Lethbridge T. : *Qualities of Relevant Software Documentation: An Industrial Study*, 2002

[Forward 02 b]
Forward A., Lethbridge T. : *The relevance of software documentation, tools and technologies : A survey* , 2002

[Glass 89]
Glass, R.: *Software maintenance documentation,* Paper presented at the Annual ACM Conference on Systems Documentation, Pittsburgh,1989.

[Hainaut 99]
Hainaut J-L. : *The DB-Main database engineering CASE tool, Version 5*, 1999

[Hainaut 00]
Hainaut J-L. : *Ingénierie des Bases de données, Deuxième Edition,* 2000

[Hartmann 01]
Hartmann J., Huang S., Tilley S. : *Documenting software systems with views II : An integrated approach based on XML*, 2001

[Holzner 02]
Holzner S. : *XSLT par la pratique*, Eyrolles, 2002

[Jahnke 99]
Jens H. Jahnke, Jörg P. Wadsack. : *Varlet : Human-centered tool support for database Reengineering* , 1999.

[Marcoux 94]
Marcoux, Y. : *Les formats normalisés de documents électroniques.*, ICO Québec, vol. 6, pp. 56-65, 1994

[Reiter 95]
Reiter E., Mellish C., Levine J. : *Automatic generation of technical documentation*, 1995

[Scott 93]
Scott R. Tilley Michael J. Whitney Hausi A. : *Personalized Information Structures* 11th International Conference on Systems Documentation, pages 325-337. ACM , 1993

[Sousa 98]
Castro Sousa M. J. , Mendes Moreira H. , *A survey on the Software Maintenance Process*, O-8186-8779-7/98 IEEE, 1998

[Stayton 03]
Stayton R. : *Using the DocBook XSL Stylesheets*, Revision 0.9, 2003

[Tryggeseth 97].
Eirik Tryggeseth : *Report from an Experiment: Impact of Documentation on Maintenance* Empirical Software Engineering, *Vol 2(2), pp. 201-207, , 1997*

[Walsh 99]
Walsh N., Muellner L. : DocBook : *The Definitive Guide, Version 1.0.2.*, O'Reilly & Associates, 1999 http://DocBook.org/tdg/en/html/DocBook.html

[Wolak 01]
Ronald G. Wolak *Software Maintenance*, 2001

OASIS (Organization for the Advancement of Structured Information Standards)
La page officielle de DocBook, où l'on trouvera les DTDs DocBook officielles, se trouve à l'adresse http://www.oasis-open.org/DocBook.html

DB-MAIN 6.5 The Database Engineering CASE environment, *Reference Manual*
Institut d'Informatique, University of Namur, http://www.db-main.be

Normes et recommandations

ISO 8879
ISO 8879:1986 (F). Traitement de l'information -- Systèmes bureautiques -- Langage normalisé de balisage généralisé (SGML): Organisation internationale de normalisation, 1986.

ISO/IEC 10744
ISO/IEC 10744-1992 (E). Technologies de l'information -- Langage de structuration temporelle/hypermédia (HyTime): Organisation internationale de normalisation, 1992.

HTML 4.01
Dave Raggett, Arnaud Le Hors, Ian Jacobs, editors. *HTML 4.01 Specification.* World Wide Web Consortium, 1999.
A l'adresse http://www.w3.org/TR/1999/REC-html401-19991224/

XHTML 1.0
Steven Pemberton, et al. *XHTML(TM) 1.0: The Extensible HyperText Markup Language.* World Wide Web Consortium, 2000.
A l'adresse http://www.w3.org/TR/2000/REC-xhtml1-20000126/
La version française : http://www.la-grange.net/w3c/xhtml1/index.html

XML 1.0
Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, and Eve Maler, editors. *Extensible Markup Language (XML) 1.0 (Second Edition).*
World Wide Web Consortium, 2000.
A l'adresse http://www.w3.org/TR/2000/REC-xml-20001006
Version française : http://babel.alis.com/web_ml/xml/REC-xml.fr.html

XSLT 1.1
James Clark, editor. *XSL Transformations.*
World Wide Web Consortium, 1999.
A l'adresse http://www.w3.org/TR/1999/REC-xslt-19991116

9. Annexes

9.1. Annexe 1 : Fonctionnement du plug-in Rep2xml.oxo

9.1.1. Introduction du paramétrage

La récupération des paramètres s'effectue via la procédure : GetParameter(string: currentpath) Cette procédure va, sur base de la position courante dans l'environnement du système :

- Appeler l'interface de paramétrage VB : parameter.exe qui permet à l'utilisateur d'introduire les paramètres souhaités et qui lors de la validation de ces paramètres va créer les fichiers de paramétrages param.txt et DocBook_param.xsl.
- Ouvrir et lire le ficher param.txt pour en récupérer les paramètres et constituer une liste des paramètres à utiliser durant la génération de la documentation.

9.1.2. Exploration du projet pour la documentation de la traçabilité

Une exploration préalable des objets du projet pour collecter les informations relative à la traçabilité est effectuée si cette documentation est demandée. Cette exploration est démarrée par l'appel de la procédure : Generate_Trace(system : sys) Cette opération permet de constitué une liste qui pour chaque objet récupère la valeur les métas propriété CurrentOID et ParentOID ainsi que la liste des objets descendants.

9.1.3. Génération du document XML neutre

La génération du document XML neutre est obtenue avec la procédure:Generer_XML (). Pour générer ce fichier XML, le plug-in explore le projet en tenant compte des d'objets sélectionnés pour être documentés. La sélection pouvant être à la fois basée sur le type d'objet et le marquage de l'objet, lors du parcours du projet on teste d'abord si le type d'objet est bien sélectionné en consultant la liste des paramètres, puis on teste si l'objet est marqué.

Sur le plan de la documentation générée pour le système et les types de productions traités, un ordre de parcours est imposé, nous explorons dans l'ordre :

- Le système dont nous extrayons les informations générales et les métas propriétés avec la procédure : Generer_MO_def(meta_object : mo)
  Le schéma data spécifique au domaine défini par l'utilisateur : procedure Generer_sch_domain(schema : sch , string: sch_type)
- Les schémas data: procedure Generer_sch_data(schema : sch , string : sch_type)
- Les schémas process : procedure Generer_sch_process(schema : sch, string: sch_type)
- Les documents : procedure Generer_doc(document : doc)

Pour l'écriture du document XML neutre, après la création du fichier au démarrage du traitement, le fichier est complété au fur et à mesure de la progression dans l'exploration du projet traité. Pour chaque type d'objet rencontré, une balise spécifique est créée et le fichier XML est édité. Pour l'écriture des attributs d'éléments XML, différentes procédures permettent d'introduire l'information avec la syntaxe XML correcte et en tenant compte du type (entier, chaîne de caractère, cardinalité, etc.)

- procedure print_att(string : att_name, string : att_value)
- procedure print_att_int(string : att_name, string : att_value)
- procedure print_cardinality(integer : min_card, integer : max_card)

9.1.4. Production des formats de publication

La production des documents est également gérée par le plug-in Rep2xml.oxo qui dans cette phase du traitement, effectue des appels aux outils de génération (parseurs XSLTproc, compilateur hhc.exe, parseurs et classes JAVA, etc.) via le fichier de script : Doc2Format.bat. Ces appels sont accompagnés de plusieurs paramètres d'exécution :

- Le format de publication à produire : XML, SingleHTML, MultiHTML,HelpHTML, PDF et RTF
- Le répertoire de départ pour le démarrage et l'exécution des outils installés.
- Le répertoire de destination des fichiers générés.
- Le nom complet du fichier du document XML neutre utilisé comme source.
- Les noms de fichiers à utiliser pour les documents produits.
- Divers paramètres spécifiques exploités lors des transformations XSLT.

Le fichier de script Doc2Format.bat

Au niveau du script Doc2Format.bat, nous avons une exécution spécifique pour chaque type de publication. Si nous prenons comme exemple les paramètres suivants :

- Le répertoire de démarrage : C:\DB-Main_DocBook\
- Le répertoire de destination : C:\Mon_DocBook\
- Le document XML neutre : Biblio_test.xml
- Le nom de fichier utilisé : Biblio_test

Nous avons pour :

- **Le format XML**

La transformation XSLT vers le DocBook par la commande :
C:\DB-Main_DocBook\xsltproc\xsltproc.exe C:\DB-Main_DocBook\DB-Main2Docbook.xsl C:\Mon_DocBook\ Biblio_test.xml > C:\Mon_DocBook\ Biblio_test_docbook.xml
La validation di DocBook par :
C:\DB-Main_DocBook\xsltproc\xmllint.exe --timing --noout --postvalid --valid *C:\Mon_DocBook\ Biblio_test_docbook.xml*
Ce format est le plus simple à générer, car il ne nécessite qu'une seule opération pour être obtenu et l'opération de validation est facultative.

- **Le format SingleHTML**

La génération et la validation du DocBook : *Biblio_test_docbook.xml*
La transformation du DocBook en page HTML unique par la commande :
C:\DB-Main_DocBook\xsltproc\xsltproc.exe C:\DB-Main_DocBook\maXSLT_HTML.xsl C:\Mon_DocBook\ Biblio_test_docbook.xml > C:\Mon_DocBook\ Biblio_test..htm

- **Le format multi HTML**

La génération et la validation du DocBook : *Biblio_test_docbook.xml*
La transformation du DocBook en un plusieurs pages HTML par la commande :
C:\DB-Main_DocBook\xsltproc\xsltproc.exe C:\DB-Main_DocBook\ maXSLT_HTMLchunk.xsl C:\Mon_DocBook\ Biblio_test_docbook.xml
Cette commande n'utilise pas de nom de fichier, mais juste le nom du répertoire de destination, la consultation du document s'effectue par le démarrage de la page Index.html générée.

- **Le format Help HTML**

La génération et la validation du DocBook : *Biblio_test_docbook.xml*
La transformation du DocBook en un plusieurs pages Help HTML par la commande :
C:\DB-Main_DocBook\xsltproc\xsltproc.exe
--stringparam htmlhelp.hhp C:\Mon_DocBook\helph\out.hhp
--stringparam htmlhelp.chm C:\Mon_DocBook\ Biblio_test .chm
C:\DB-Main_DocBook\ maXSLT_HHTML.xsl C:\Mon_DocBook\ Biblio_test_docbook.xml
Cette commande comprend deux paramètres spécifiques à la transformation XSLT et qui sont relatifs au nom du projet et au nom du fichier de sortie HelpHTML.
La conversion des pages HelpHTML en un fichier d'aide de type .chm par la commande :
C:\DB-Main_DocBook \HTMLHelp\hhc C:\Mon_DocBook\helph\out.hhp

Remarque :
Pour ce format, nous avons également la présence plusieurs commandes DOS qui permettent la création puis l'effacement du répertoire intermédiaire : *C:\Mon_DocBook\helph* ce qui permet l'effacement automatique des pages Help HTML qui sont devenues inutiles.

- **Le format PDF**

La génération et la validation du DocBook : *Biblio_test_docbook.xml*
La génération du fichier intermédiaire FO à partir du DocBook par la commande :
C:\DB-Main_DocBook\xsltproc\xsltproc.exe *C:\DB-Main_DocBook*\maXSLT_FO.xsl
C:\Mon_DocBook\ Biblio_test_docbook.xml > C:\Mon_DocBook\ Biblio_test..fo
L'appel de l'outil de conversion JAVA FOP qui génère le fichier PDF par la commande :
java org.apache.fop.apps.Fop C:\Mon_DocBook\ Biblio_test..fo C:\Mon_DocBook\ Biblio_test.pdf

Remarque :
L'appel de l'outil JAVA est précédé d'un ensemble de commandes permettant de définir les chemins complets des classes JAVA utilisées.

- **Le format RTF**

La génération et la validation du DocBook : *Biblio_test_docbook.xml*
La génération du fichier intermédiaire FO à partir du DocBook par la commande :
C:\DB-Main_DocBook\xsltproc\xsltproc.exe *C:\DB-Main_DocBook*\maXSLT_FO.xsl
C:\Mon_DocBook\ Biblio_test_docbook.xml > C:\Mon_DocBook\ Biblio_test..fo
L'appel de l'outil de conversion JAVA XMLMINF FO qui génère le fichier RTF par :
java com.xmlmind.fo.converter.Driver C:\Mon_DocBook\ Biblio_test..fo
C:\Mon_DocBook\ Biblio_test.rtf

Remarque :
Ici également, un ensemble de commandes permettent de définir les chemins complets des classes JAVA utilisées.

9.2. Annexe 2 : DTD des documents XML neutre

La DTD des documents XML neutres : DB-Main_XML.dtd est détaillée ci-dessous.

Pour la facilité de lecture, les éléments sont présentés dans l'ordre alphabétique et chaque élément est accompagné de sa liste d'attributs lorsque ces attributs existent.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD DB-Main XML document 2003 -->

<!ELEMENT attribute (ATT_si | ATT_comp)>

<!ELEMENT ATT_comp (description*, note*, meta_prop, trace?, proc_unit_use*, attribute+)>

<!ATTLIST ATT_comp
    name CDATA #REQUIRED
    short_name CDATA #IMPLIED
    Id_att CDATA #REQUIRED
    cardinality CDATA #REQUIRED
    set_type CDATA #IMPLIED
    type CDATA #REQUIRED>

<!ELEMENT ATT_si (description*, note*, trace?, meta_prop, proc_unit_use*)>
<!ATTLIST ATT_si
    name CDATA #REQUIRED
    short_name CDATA #IMPLIED
    Id_att CDATA #REQUIRED
    cardinality CDATA #REQUIRED
    set_type CDATA #IMPLIED
    type CDATA #REQUIRED
    length CDATA #IMPLIED
    decim CDATA #IMPLIED
    stable CDATA #REQUIRED
    recyclable CDATA #REQUIRED
    Dom_name CDATA #IMPLIED
    Id_Dom CDATA #IMPLIED>

<!ELEMENT cluster (sub_entity)>
<!ATTLIST cluster
    name CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT collection (description*, note*, meta_prop, entite_collection)>
<!ATTLIST collection
    name CDATA #REQUIRED
    Id_col CDATA #REQUIRED>

<!ELEMENT collections (#PCDATA | collection)*>

<!ELEMENT description (#PCDATA | description_type)*>
```

```
<!ELEMENT description_type EMPTY>
<!ATTLIST description_type
    type CDATA #REQUIRED>

<!ELEMENT doc_texte (line+)>

<!ELEMENT dom_entities (dom_entity_type)>

<!ELEMENT dom_entity_type (attribute*)>
<!ATTLIST dom_entity_type
    name CDATA #REQUIRED
    short_name CDATA #IMPLIED
    Id_ent CDATA #REQUIRED>

<!ELEMENT enti_collec EMPTY>
<!ATTLIST enti_collec
    name CDATA #REQUIRED
    Id_ent CDATA #REQUIRED>

<!ELEMENT entite EMPTY>
<!ATTLIST entite
    name CDATA #REQUIRED
    Id_ent CDATA #REQUIRED>

<!ELEMENT entite_collection (enti_collec+)>

<!ELEMENT entite_relation (relation, entite*)>
<!ATTLIST entite_relation
    name CDATA #REQUIRED>

<!ELEMENT entities (entity_type+)>

<!ELEMENT entity_type (description*, note*, meta_prop, trace?, cluster?, is_in_cluster?, attribute*,
group*, entite_relation*)>
<!ATTLIST entity_type
    name CDATA #REQUIRED
    short_name CDATA #IMPLIED
    Id_ent CDATA #REQUIRED>

<!ELEMENT et_role EMPTY>
<!ATTLIST et_role
    name CDATA #REQUIRED
    Id_ent CDATA #REQUIRED>

<!ELEMENT file (meta_prop*, doc_texte)>
<!ATTLIST file
    name CDATA #REQUIRED
    version CDATA #REQUIRED
    type_of_file CDATA #IMPLIED
    path CDATA #REQUIRED
    date CDATA #REQUIRED
    modif_date CDATA #REQUIRED
    Id_doc CDATA #REQUIRED>
```

```
<!ELEMENT group (RELATION_def*, (GR_comp_role | GR_comp_att)*, description*, note*, trace?,
meta_prop?)>
<!ATTLIST group
    name CDATA #REQUIRED
    Id_group CDATA #REQUIRED
    cardinality CDATA #REQUIRED
    identifier CDATA #REQUIRED
    access_key CDATA #REQUIRED
    coexistence CDATA #REQUIRED
    atleastone CDATA #REQUIRED
    exclusive CDATA #REQUIRED>


<!ELEMENT GR_comp_att EMPTY>
<!ATTLIST GR_comp_att
    name CDATA #REQUIRED>

<!ELEMENT GR_comp_role EMPTY>
<!ATTLIST GR_comp_role
    name CDATA #REQUIRED
    rt_name CDATA #REQUIRED>

<!ELEMENT ln (#PCDATA)>

<!ELEMENT internal_data_object (#PCDATA | attribute)*>

<!ELEMENT is_in_cluster (sup_entity)>
<!ATTLIST is_in_cluster
    name CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT line (ln)>
<!ATTLIST line
    num_line CDATA #REQUIRED>

<!ELEMENT meta_prop (#PCDATA | MP_val)*>

<!ELEMENT MO_def (#PCDATA | MP_def)*>
<!ATTLIST MO_def
    name CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT MP_def (#PCDATA | description)*>
<!ATTLIST MP_def
    name CDATA #REQUIRED
    value_type CDATA #REQUIRED
    updatable CDATA #REQUIRED
    multivalued CDATA #REQUIRED
    predefined CDATA #REQUIRED
    hidden CDATA #REQUIRED>

<!ELEMENT MP_val (MP_value)>
<!ATTLIST MP_val
    name CDATA #REQUIRED
    oid CDATA #IMPLIED>
```

```
<!ELEMENT MP_value (#PCDATA)>

<!ELEMENT note (#PCDATA)>

<!ELEMENT parent_OID EMPTY>
<!ATTLIST parent_OID
    type CDATA #REQUIRED
    parent_Name CDATA #REQUIRED
    parent_OID CDATA #REQUIRED>

<!ELEMENT proc_data_object EMPTY>
<!ATTLIST proc_data_object
    name CDATA #REQUIRED
    Id_do CDATA #REQUIRED
    mode CDATA #REQUIRED>

<!ELEMENT proc_unit (description?, meta_prop, proc_data_object*, proc_unit_call*, proc_data_object*,
proc_unit_decomp*, proc_unit_comp?)>
<!ATTLIST proc_unit
    name CDATA #REQUIRED
    type CDATA #REQUIRED
    Id_pu CDATA #REQUIRED
    short_name CDATA #IMPLIED>

<!ELEMENT proc_unit_call EMPTY>
<!ATTLIST proc_unit_call
    name CDATA #REQUIRED
    Id_puc CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT proc_unit_comp EMPTY>
<!ATTLIST proc_unit_comp
    name CDATA #REQUIRED
    Id_puc CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT proc_unit_decomp EMPTY>
<!ATTLIST proc_unit_decomp
    name CDATA #REQUIRED
    Id_pud CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT proc_unit_use EMPTY>
<!ATTLIST proc_unit_use
    name CDATA #REQUIRED
    Id_puu CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT proc_units (proc_unit+)>

<!ELEMENT rel_type (description*, note*, meta_prop, trace?, role+, attribute*, group?)>
<!ATTLIST rel_type
    name CDATA #REQUIRED
    short_name CDATA #IMPLIED
    Id_rel CDATA #REQUIRED>
```

```
<!ELEMENT RELATION_def EMPTY>
<!ATTLIST RELATION_def
    type CDATA #REQUIRED
    name CDATA #REQUIRED
    Id_Target CDATA #REQUIRED
    parent_name CDATA #REQUIRED
    Id_Parent_Target CDATA #REQUIRED
    parent_type CDATA #REQUIRED>

<!ELEMENT relation EMPTY>
<!ATTLIST relation
    name CDATA #REQUIRED
    Id_rel CDATA #REQUIRED>

<!ELEMENT relations (#PCDATA | rel_type)*>

<!ELEMENT role (et_role, description*, meta_prop)>
<!ATTLIST role
    cardinality CDATA #REQUIRED
    name CDATA #IMPLIED>

<!ELEMENT schema_data (description*, note*, meta_prop, entities, relations, collections)>
<!ATTLIST schema_data
    name CDATA #REQUIRED
    version CDATA #REQUIRED
    short_name CDATA #IMPLIED
    date CDATA #REQUIRED
    modif_date CDATA #REQUIRED
    Id_sch CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT schema_dom (dom_entities)>
<!ATTLIST schema_dom
    name CDATA #REQUIRED
    version CDATA #REQUIRED
    short_name CDATA #IMPLIED
    date CDATA #REQUIRED
    modif_date CDATA #REQUIRED
    Id_sch CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT schema_process (description*, note*, meta_prop*, proc_units*, internal_data_object*)>
<!ATTLIST schema_process
    name CDATA #REQUIRED
    version CDATA #REQUIRED
    short_name CDATA #IMPLIED
    date CDATA #REQUIRED
    modif_date CDATA #REQUIRED
    Id_sch CDATA #REQUIRED
    type CDATA #REQUIRED>

<!ELEMENT shild_OID EMPTY>
<!ATTLIST shild_OID
    type CDATA #REQUIRED
    shild_Name CDATA #REQUIRED
    shild_OID CDATA #REQUIRED>
```

```
<!ELEMENT sub_enti EMPTY>
<!ATTLIST sub_enti
    name CDATA #REQUIRED
    Id_ent CDATA #REQUIRED>

<!ELEMENT sub_entity (sub_enti+)>

<!ELEMENT sup_entity EMPTY>
<!ATTLIST sup_entity
    name CDATA #REQUIRED
    Id_ent CDATA #REQUIRED>

<!ELEMENT system (description*, MO_def+, meta_prop*, schema_dom*, schema_data*,
schema_process*, file*)>
<!ATTLIST system
    name CDATA #REQUIRED
    short_name CDATA #IMPLIED
    Schema_creation_date: CDATA #REQUIRED
    Documentation_Creation_Date: CDATA #REQUIRED>

<!ELEMENT trace (#PCDATA | parent_OID | shild_OID)*>
```

9.3. Annexe 3 : Installation et utilisation du prototype DB-Main DocBook

Le prototype a été développé et testé pour l'OS Windows2000 et NT4.

9.3.1. Installation et environnement d'exécution

Les fichiers nécessaires au fonctionnement de l'outil doivent être situés dans un répertoire dédié, de manière à présenter l'arborescence suivante :

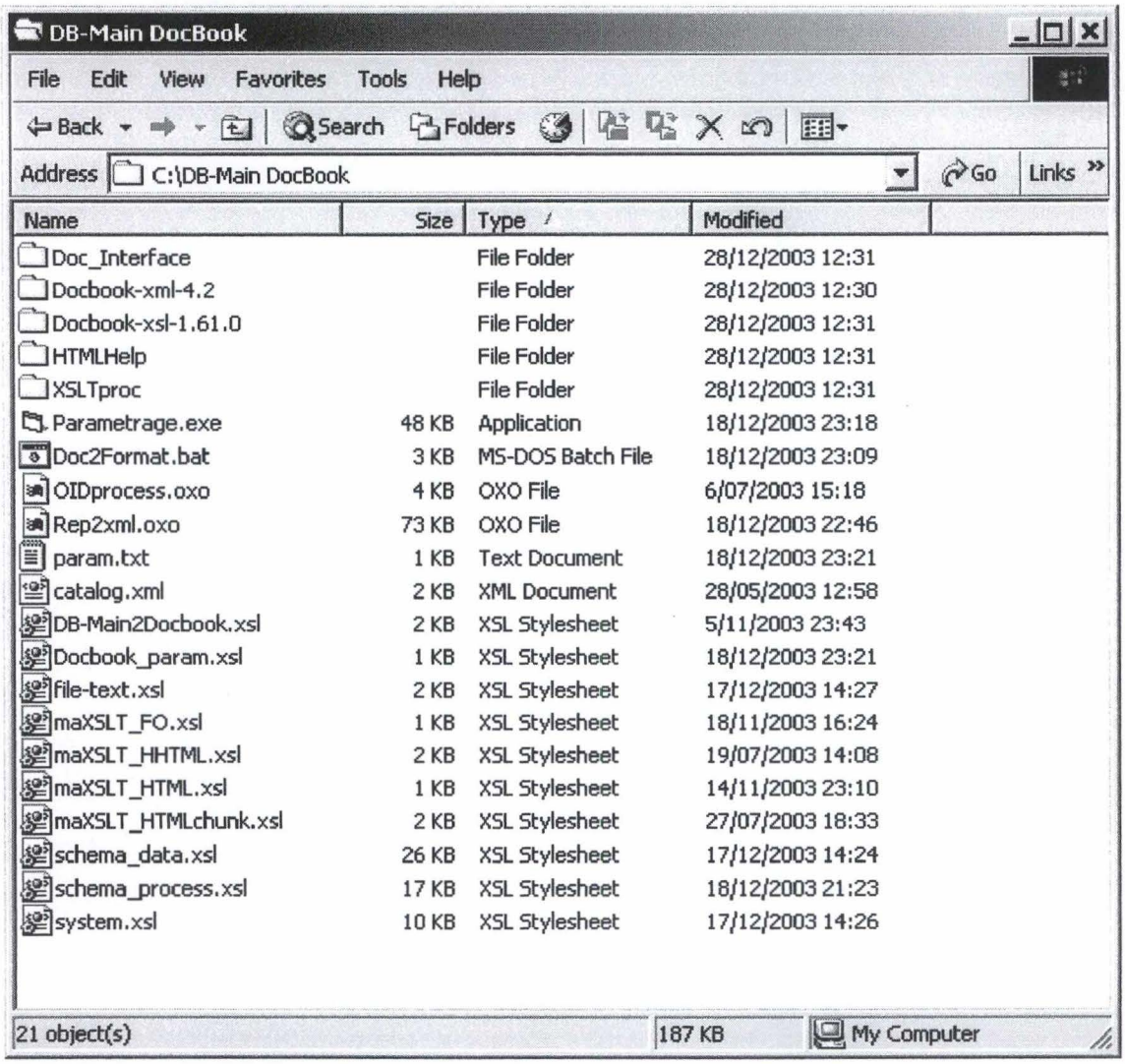

Figure 19 Répertoire reprenant les composants de l'outil

Sur le plan de l'environnement d'exécution, c'est la position du plug-in principal Rep2xml.oxo qui détermine la position de l'arborescence et c'est par rapport à ce fichier que les composants doivent être correctement situés.

Nous retrouvons les composants suivants :

9.3.1.1. Les répertoires

- Doc_interface
  Ce répertoire contient le package d'installation de l'interface VB avec tous les composants nécessaires. **L'interface VB de l'outil doit elle être installée avec le fichier d'installation SETUP.exe qui se situe dans ce répertoire afin que les composants soient correctement installés.**

- DocBook-xml-4.2
  Le répertoire DocBook-xml-4.2 contient la DTD de DocBook pour sa version 4.2. Ce répertoire est exploité lors de l'opération de validation des DocBooks créés.

- DocBook-xsl-1.61.0
  Répertoire reprenant les stylesheets DocBook version 1.61.0 permettant la transformation des documents XML DocBooks vers les formats HTML (single, multi et Help HTML) et vers le format FO utilisé pour obtenir les formats d'impression (PDF, RTF et PS).

- HTMLHelp
  C'est dans ce répertoire que se trouve le compilateur hhc.exe qui rassemble et comprime dans un seul fichier de type .chm, les pages HTML générées en mode HelpHTML

- XSLTproc
  Le répertoire XSLTproc contient les parseurs rapides écrits en C. Nous avons dans ce répertoire le parseur de transformation XSLTproc.exe et le parseur de validation XMLlint.exe ainsi que toutes les librairies nécessaires.

9.3.1.2. Les fichiers d'exécution

- Rep2xml.oxo
  Le plug-in Rep2xml.oxo permet de démarrer l'outil de documentation depuis l'atelier DB-Main. C'est ce plug-in qui gère l'ensemble du traitement.

- Parametrage.exe
  C'est le fichier d'exécution de l'interface de paramétrage. L'interface de paramétrage VB permet, comme nous le verrons plus loin, d'introduire les paramètres utilisés pour la génération de la documentation.

- Doc2format.bat
  Ce fichier de script est appelé par le plug-in Rep2xml.oxo lorsque l'outil génère les différents types de document. L'appel à ce script s'accompagne de plusieurs paramètres qui vont déterminer quels seront les outils mis en œuvre en fonction du type de format généré.

- OIDprocess.oxo
  C'est le plug-in utilisé pour documenter la traçabilité forward/backward. Il est appelé durant le développement pour créer et transmettre les métas propriétés qui permettent de tracer les objets de schéma en schéma.

9.3.1.3. Les stylesheets DB-Main

Pour ces stylesheets, nous avons d'une part, les stylesheets permettant la transformation du document XML neutre vers le document XML DocBook :

- DB-Main2DocBook.xsl
- System.xsl
- Schema_data.xsl
- Schema_process.xsl
- File_text.xsl

Et d'autre part, les stylesheets utilisées pour les transformations vers les formats de publication :

- maXSLT_HTML.xsl pour générer une page HTML de type single HTML
- maXSLT_HTMLchunk.xsl pour générer les pages HTML pour le multi HTML
- maXSLT_HHTML.xsl pour générer les pages HTML pour le format Help HTML
- maXSLT_FO.xsl pour obtenir le fichier Formating Object destiné aux formats d'impression.

9.3.1.4. Les fichiers de paramétrage et le fichier de résolution d'adresse de la DTD DocBook

Les fichiers de paramétrage sont des fichiers générés automatiquement par l'interface de paramétrage de l'outil. Lorsque l'utilisateur valide ses choix, nous avons la création de deux fichiers :

- Param.txt
  C'est un fichier texte qui indique à l'application quels sont les types d'objet à documenter ainsi que d'autres sélections tels que le souhait d'obtenir la documentation de la traçabilité. Il s'agit donc de paramètres relatifs au contenu de la documentation.
- DocBook_parm.xsl
  Cette stylesheet introduit les paramètres relatifs à la présentation. La stylesheet DocBook_param.xsl est importée avec les autres stylesheets durant les opérations de transformation XSLT.

Le fichier de résolution de l'adresse de localisation de la DTD, catalog.xml, fournit au parseur de validation XMLlint.exe une adresse locale pour la DTD DocBook utilisée. Ceci permet l'accès à cette DTD sans devoir utiliser le réseau Internet.

Remarque :

Pour que l'opération de validation puisse s'exécuter correctement avec un adressage local de la DTD, le fichier catalog.xml doit être édité de manière à faire correspondre le paramètre *« base»* de l'entité group avec la position effective du répertoire de la DTD DocBook-xml-4.2.
Pour l'exemple illustré ci-dessus, nous avons dans le fichier catalog.xml le paramètre base suivant :

<!-- DTD files installed under -->
<group xml:base="file:///c:/DB-Main Docbook/ " >

9.3.2. Utilisation du prototype

L'utilisation s'effectue directement depuis l'interface de DB-Main par l'appel du plug-in Voyager : Rep2xml.oxo

9.3.2.1. Appel de l'interface de paramétrage

Une fois démarré, ce plug-in fait apparaître automatiquement l'interface de paramétrage de l'outil qui permet la sélection du format de documentation à générer ainsi que les paramètres utilisés.

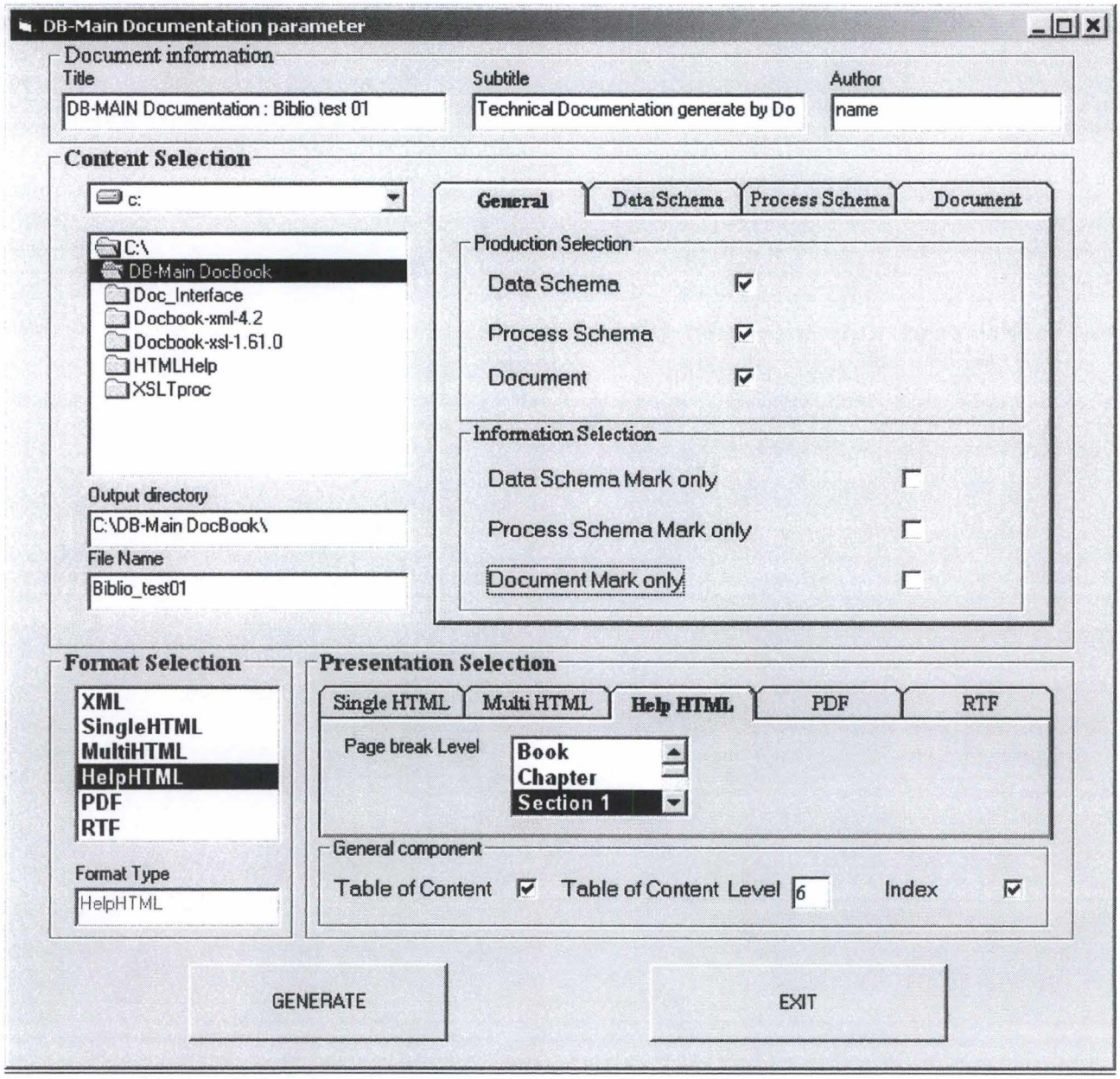

Figure 20 Interface de paramétrage de l'outil

Après avoir introduit les éléments d'information générale sur le document à générer : titre, sous-titre et auteur, l'utilisateur est invité à sélectionner les paramètres pour la génération de la documentation.

9.3.2.2. Sélection du contenu

Le paramétrage du contenu à documenter est effectué via les 4 onglets de sélection :

1. General :
   L'onglet General permet une sélection des productions à documenter.
   - Soit par type de production : schémas data, schémas process et documents
   - Soit par marquage des productions. Cette seconde possibilité implique qu'une sélection des productions avec la fonction Mark de l'atelier DB-Main a été effectuée au préalable.

2. Data Schema :
   Cet onglet permet de sélectionner les objets relatifs aux schémas data d'un projet ici aussi deux types de sélection sont possible.
   - Soit par type d'objet : types d'entité, types d'association, collections, attributs, groupes et métas propriétés.
   - Soit par marquage avec l'option : *« Object with mark only »*

   Deux autres sélections sont également disponibles pour les schémas data :
   - La sélection *« Forward/Backward information »* qui permet de demander la documentation de la traçabilité.
   - La sélection *« Note »* qui permet de documenter les notes attachées aux objets.

3. Process Schema :
   Pour les schémas process, nous utilisons la même approche que pour la sélection du contenu des schémas process. Les types d'objet que l'on peut sélectionner sont ici : les unités process, les objets de données internes, les objets de données externes, les relations de type appel et les relations de type décomposition.
   Nous retrouvons également une option qui permet de limiter la documentation aux objets marqués et la possibilité de documenter les notes. Nous n'avons pas de documentation de la traçabilité pour les schémas process.

4. Document :
   L'onglet « Document » permet uniquement une sélection sur base du type de document : fichiers texte, rapport, fichiers .ddl, fichiers COBOL, fichiers CODASYL, etc.
   Pour la documentation des documents, une option de numérotage des lignes du texte est proposée.

Remarque :

L'implémentation de la sélection du contenu est encore à finaliser pour la partie schéma process et la partie document, les options de sélection pour ces ongles ne sont donc pas activées.

9.3.2.3. Sélection de la présentation

Les paramètres relatifs à la présentation et à la mise en forme des documents sont particulièrement nombreux dans le modèle DocBook. Pour le développement du prototype de l'outil, nous nous sommes limités à proposer quelques un parmi les plus importants :
Pour les paramètres applicables à tous les formats de publication, nous avons :

- La présence ou l'absence d'une table des matières.
- Le niveau de détail des sous-sections de cette table des matières.
- La présence ou l'absence d'un index.

Pour les paramètres spécifiques aux formats mettant en œuvre plusieurs pages HTML, nous proposons de permettre à l'utilisateur de sélectionner le niveau de découpage du document en pages HTML. Les niveaux de découpage proposés correspondent aux différents niveaux hiérarchiques du modèle du livre proposé par DocBook avec la création d'une nouvelle page au niveau des chapitres, des sections, des sous-sections, etc..

9.3.2.4. Sélection du format

La sélection du format de publication propose les possibilités suivantes :

- XML pour un simple document XML DocBook.
- Single HTML pour une documentation sur une seule page HTML pour les petites documentations.
- Multi HTML, c'est le format de documentation de base qui permet de consulter les informations à travers un ensemble de pages HTML dans les quelles on navigue depuis une table des matières ou via les liens présents dans ces pages.
- Help HTML permet générer un fichier d'aide de type .chm qui permet de visualiser les pages HTML de la documentation au travers du navigateur d'aide de Microsoft.
- PDF pour la génération d'une documentation dans le format de publication bien connu de l'éditeur Adobe.
- RTF qui permet de générer une documentation exploitable par la plupart des applications de traitement de texte.

Remarque :
Les formats de publication PDF et RTF ne sont que partiellement implémentés dans le prototype du générateur et requièrent l'installation d'un répertoire additionnel contenant les outils et les classes JAVA nécessaires à leur production.

9.3.2.5. Validation des paramètres et génération de la documentation

Après la sélection des paramètres, la sélection du répertoire de sortie et l'introduction du nom de fichier à utiliser (nom de fichier sans extension), on peut valider ces informations et démarrer le traitement en appuyant sur le bouton GENERATE. Le démarrage du traitement s'accompagne de l'affichage de la console de Voyager ainsi que d'une console DOS qui permettent de suivre l'exécution.

Le Bouton EXIT permet simplement de quitter l'outil sans générer de documentation.

9.4. Annexe 4 : Exemple de documentation générée : Le projet BIBLIO

Nous présentons ici la documentation générée pour le schéma conceptuel : BIBLIO
Pour cet exemple nous présentons :

- Le schéma conceptuel à documenter tel que nous le trouvons dans l'interface graphique de l'atelier DB-Main.
- Deux extraits de la documentation de type Help HTML générée pour la consultation par navigation.
- Le document au format RTF généré qui reprend la documentation pour ce schéma conceptuel ainsi que la documentation d'un fichier texte.

9.4.1. Le Schéma conceptuel BIBLIO

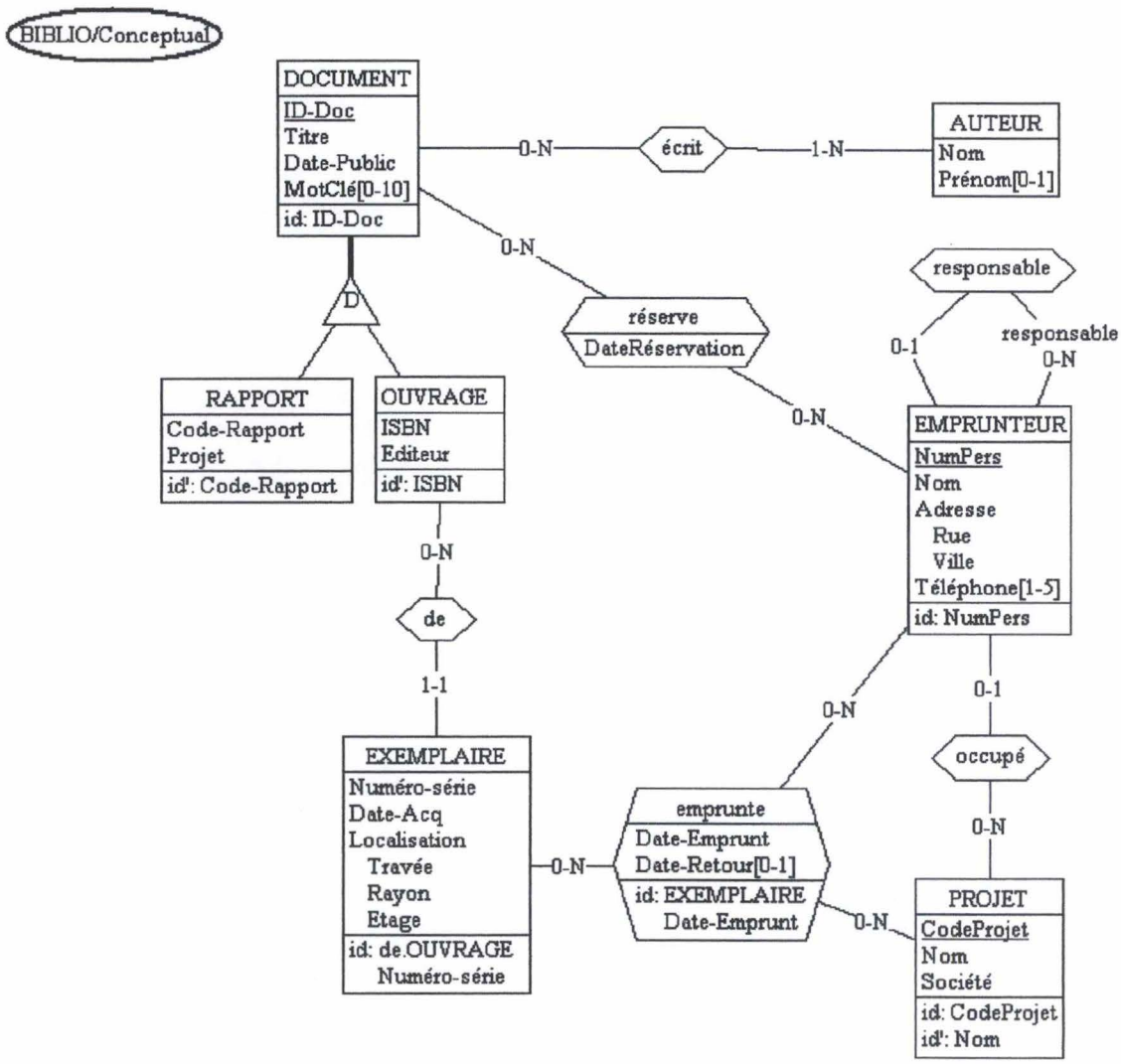

Figure 21 Schéma conceptuel de BIBLIO en mode graphique

9.4.2. Extrait de la documentation de type Help HTML

1. Le premier extrait de la documentation de type Help HTML montre les informations relatives au type d'entité AUTEUR

   Pour les types d'entité, les éléments documentés sont :
   - Les informations générales : Nom, nom court, identifiant technique
   - Les descriptions sémantiques et techniques.
   - Les clusters
   - Les attributs avec leurs détails : Nom, nom court, type, cardinalité, etc.
   - Les types d'association impliqués avec ce type d'entité.
   - Les groupes
   - Les métas propriétés

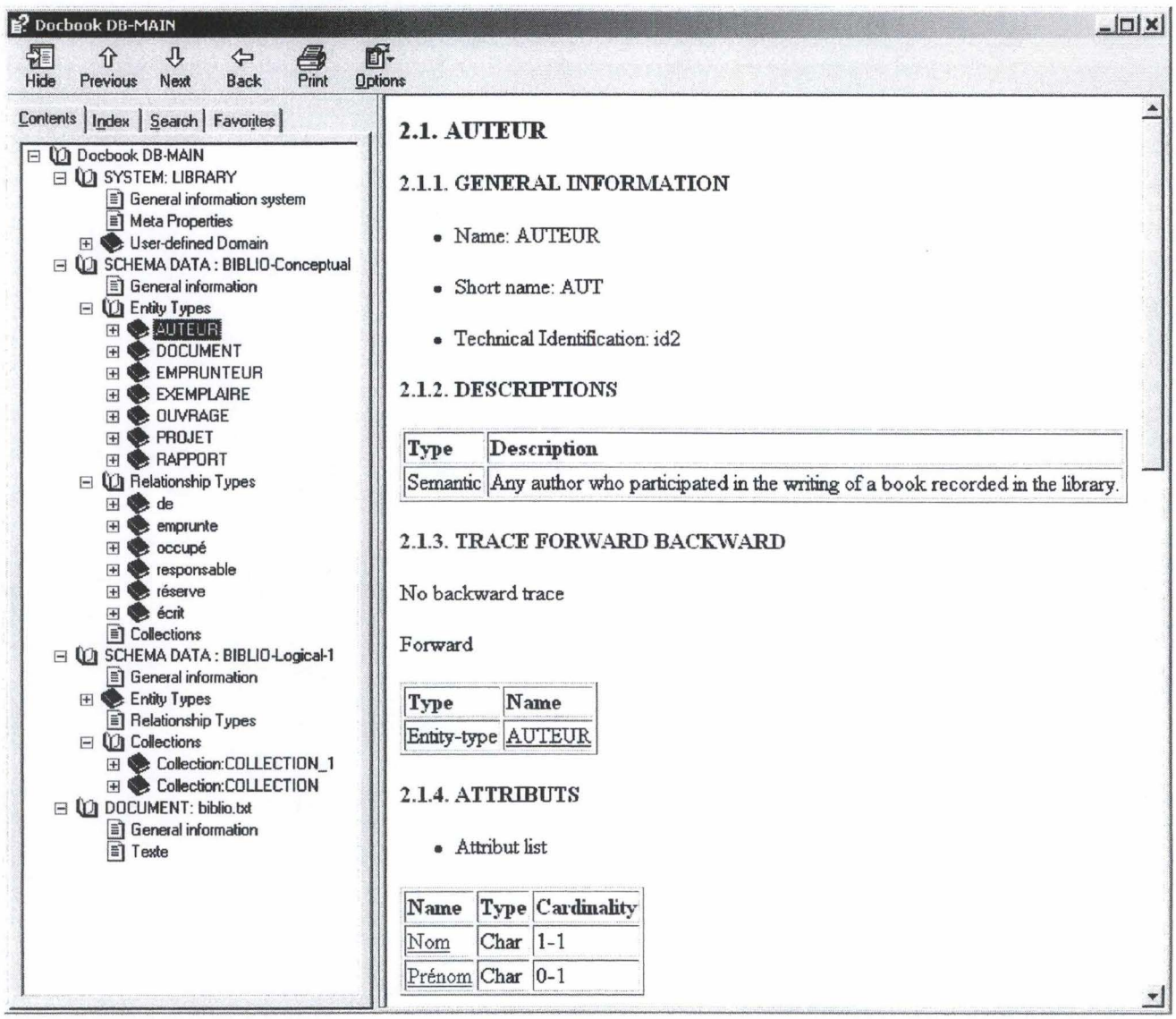

Figure 22 Extrait de la documentation Help HTML : type d'entité AUTEUR

2. Le second extrait de la documentation de type Help HTML montre les informations relatives au type d'association : emprunte

   Pour les types d'association, les éléments documentés sont :
   - Les informations générales : Nom, nom court, identifiant technique
   - Les descriptions sémantiques et techniques
   - Les attributs avec leurs détails : Nom, nom court, type, cardinalité, etc.
   - Les rôles
   - Les groupes
   - Les métas propriétés

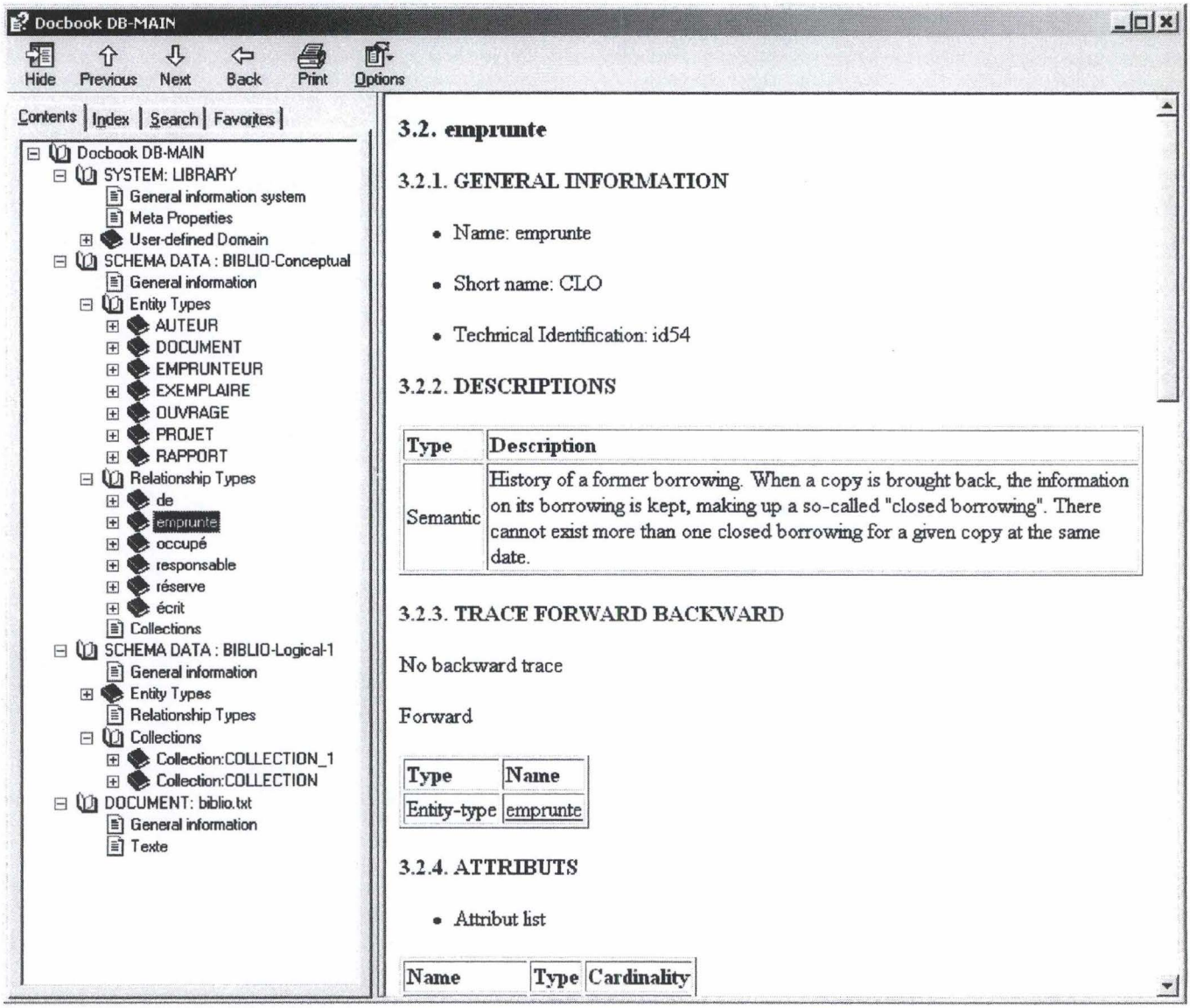

Figure 23 Extrait de la documentation Help HTML : type d'association emprunte

9.4.3. Exemple de documentation de type RTF

Technical Documentation generate by DocBook

Docbook DB-MAIN

DB-MAIN Documentation : BIBLIO Exemple

Jacques Furnelle

Published Creation date: 1-1-2004

Chapter 1. SYSTEM: LIBRARY

1. General information system

- Name: LIBRARY
- Short name:
- System creation date : 22-06-1994

Table 1.1. DESCRIPTION

Type	Description
Technical	#GHSPosX=846 #GHSPosY=52

Table 1.2. DOCUMENT LIST

Name	Version	Type of file	Date	Modif Date
biblio.txt	Texte1	texte	31-12-2003	31-12-2003

2. Meta Properties

Table 1.3. Meta object : Decomposition

Meta property	Type	Updatable	Multivalued	Predefined
Stereotype	string	Yes	Yes	Yes

Table 1.4. Meta object : Call

Meta property	Type	Updatable	Multivalued	Predefined
Stereotype	string	Yes	Yes	Yes

Table 1.5. Meta object : In-out

Meta property	Type	Updatable	Multivalued	Predefined
Stereotype	string	Yes	Yes	Yes

Table 1.6. Meta object : Collection

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No
Stereotype	string	Yes	Yes	Yes

Table 1.7. Meta object : Role

Meta property	Type	Updatable	Multivalued	Predefined
Stereotype	string	Yes	Yes	Yes
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No

Table 1.8. Meta object : Group

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No

Table 1.9. Meta object : Compound attribute

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No
Stereotype	string	Yes	Yes	Yes

Table 1.10. Meta object : Atomic attribute

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No
Default value	string	Yes	No	No
Value constraint	string	Yes	Yes	No
Stereotype	string	Yes	Yes	Yes

Table 1.11. Meta object : Rel-type

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No
Stereotype	string	Yes	Yes	Yes

Table 1.12. Meta object : Entity type

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No
Stereotype	string	Yes	Yes	Yes

Table 1.13. Meta object : Schema

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No
IsView	boolean	No	No	No

Table 1.14. Meta object : Processing unit

Meta property	Type	Updatable	Multivalued	Predefined
Parent_OID	num	Yes	No	No
Copy_OID	num	Yes	No	No
Stereotype	string	Yes	Yes	Yes

3. User-defined Domain

- Domain list

Name	Type	Cardinality
test domain	Char	1-1

3.1. User domain: test domain

Name	Short Name	Type	Card	Set Type	Stable	Recycl
test domain	test	Char	1-1	set	No	Yes

Chapter 2. SCHEMA DATA : BIBLIO-Conceptual

1. General information

- Name: BIBLIO
- Short name: LIB
- Identifiant: id1

Table 2.1. DESCRIPTION

Type	Description
Semantic	Variante de la base de données BIBLIO. Inclut une structure IS-A.
Technical	#GHSPosX=211 #GHSPosY=144 #zoom=75 #GraphCouple=1

Table 2.2. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	1
IsView	0

2. Entity Types

2.1. AUTEUR

2.1.1. GENERAL INFORMATION

- Name: AUTEUR
- Short name: AUT
- Technical Identification: id2

2.1.2. DESCRIPTIONS

Type	Description
Semantic	Any author who participated in the writing of a book recorded in the library.

2.1.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	AUTEUR

2.1.4. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
Nom	Char	1-1
Prénom	Char	0-1

2.1.4.1. Nom

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Nom		Char	1-1		No	Yes	Entity: AUTEUR

2.1.4.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Nom

2.1.4.2. Prénom

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Prénom		Char	0-1		No	Yes	Entity: AUTEUR

2.1.4.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Prénom

2.1.5. RELATIONSHIP TYPES

- Relationship types list

Name
écrit

2.1.5.1. écrit

Entity in relation
DOCUMENT

2.1.6. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	id2

2.2. DOCUMENT

2.2.1. GENERAL INFORMATION

- Name: DOCUMENT
- Technical Identification: id897
- NOTE: NOTE sur document

2.2.2. DESCRIPTIONS

Type	Description
Semantic	Commentaire
Technical	technique

2.2.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	DOCUMENT

2.2.4. CLUSTERS

- Name: DOCUMENT
- Type:Disjoint

Subtype
OUVRAGE
RAPPORT

2.2.5. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
ID-Doc	Char	1-1
Titre	Char	1-1
Date-Public	Date	1-1
MotClé	Char	0-10

2.2.5.1. ID-Doc

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
ID-Doc		Char	1-1		No	Yes	Entity: DOCUMENT

2.2.5.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	ID-Doc
Simple Attribut	ID-Doc
Simple Attribut	ID-Doc
Simple Attribut	ID-Doc
Simple Attribut	ID-Doc
Simple Attribut	ID-Doc
Simple Attribut	ID-Doc
Simple Attribut	ID-Doc
Simple Attribut	OUVRAGE
Simple Attribut	RAPPORT

2.2.5.2. Titre

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Titre		Char	1-1		No	Yes	Entity: DOCUMENT

2.2.5.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Titre

2.2.5.3. Date-Public

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Date-Public		Date	1-1		No	Yes	Entity: DOCUMENT

2.2.5.3.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Date-Public

2.2.5.4. MotClé

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
MotClé		Char	0-10	set	No	Yes	Entity: DOCUMENT
Description: Key words describing the themes or the style of the book.							

2.2.5.4.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	MotClé

2.2.6. RELATIONSHIP TYPES

- Relationship types list

Name
réserve
écrit

2.2.6.1. réserve

Entity in relation
EMPRUNTEUR

2.2.6.2. écrit

Entity in relation
AUTEUR

2.2.7. GROUPS

2.2.7.1. Group: IDDOCUMENT

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
ID-Doc	0-1	Primary	No	No	No	No

2.2.7.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	IDDOCUMENT

2.2.8. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	id897

2.3. EMPRUNTEUR

2.3.1. GENERAL INFORMATION

- Name: EMPRUNTEUR
- Short name: BER
- Technical Identification: id15

2.3.2. DESCRIPTIONS

Type	Description
Semantic	Any person employed in a company affiliated to the library, and therefore authorized to borrow books.

2.3.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	EMPRUNTEUR

2.3.4. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
NumPers	Char	1-1
Nom	Char	1-1
Adresse	Compound	1-1
Téléphone	Numeric	1-5

2.3.4.1. NumPers

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
NumPers		Char	1-1		No	Yes	Entity: EMPRUNTEUR

2.3.4.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	NumPers
Simple Attribut	NumPers
Simple Attribut	NumPers
Simple Attribut	NumPers
Simple Attribut	Responsable

2.3.4.2. Nom

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Nom		Char	1-1		No	Yes	Entity: EMPRUNTEUR

2.3.4.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Nom

2.3.4.3. Adresse

Name	Short Name	Type	Card	Set Type	Owner
Adresse		Compound	1-1		Entity: EMPRUNTEUR
Description: Contact address of the borrower.					

- Compound attribut elements

Name	Type	Cardinality
Rue	Char	1-1
Ville	User Domain	1-1

2.3.4.3.1. Rue

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Rue		Char	1-1		No	Yes	Comp att.:Adresse

2.3.4.3.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Adr_Rue

2.3.4.3.2. Ville

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Ville		User Domain:test domain [2]	1-1	set	Yes	No	Comp att.:Adresse

2.3.4.3.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Adr_Ville

2.3.4.4. Téléphone

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Téléphone		Numeric	1-5	set	No	Yes	Entity: EMPRUNTEUR
Description: Phone numbers at which the borrower can be contacted.							

2.3.4.4.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Téléphone

2.3.5. RELATIONSHIP TYPES

- Relationship types list

Name
occupé
réserve
responsable
responsable
emprunte

2.3.5.1. occupé

Entity in relation
PROJET

2.3.5.2. réserve

Entity in relation
DOCUMENT

2.3.5.3. responsable

Entity in Relation : EMPRUNTEUR

2.3.5.4. responsable

Entity in Relation : EMPRUNTEUR

2.3.5.5. emprunte

Entity in relation
EXEMPLAIRE
PROJET

2.3.6. GROUPS

2.3.6.1. Group: 1

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
NumPers	0-1	Primary	No	No	No	No

2.3.6.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	1

2.3.7. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	id15

2.4. EXEMPLAIRE

2.4.1. GENERAL INFORMATION

- Name: EXEMPLAIRE
- Short name: COPY
- Technical Identification: id26

2.4.2. DESCRIPTIONS

Type	Description
Semantic	A copy is an instance of a book. A book can have an arbitrary number of copies in the library.

2.4.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	EXEMPLAIRE

2.4.4. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
Numéro-série	Numeric	1-1
Date-Acq	Date	1-1
Localisation	Compound	1-1

2.4.4.1. Numéro-série

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Numéro-série		Numeric	1-1		No	Yes	Entity: EXEMPLAIRE
Description: Serial number of the copy. Identifies it among all the copies of the same book.							

2.4.4.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Numéro-série
Simple Attribut	Numéro-série **[Erreur! Signet non défini.]**

2.4.4.2. Date-Acq

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Date-Acq		Date	1-1		No	Yes	Entity: EXEMPLAIRE
Description: Date on which this copy was received.							

2.4.4.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Date-Acq

2.4.4.3. Localisation

Name	Short Name	Type	Card	Set Type	Owner
Localisation		Compound	1-1		Entity: EXEMPLAIRE
Description: Physical location of the copy in the library.					

- Compound attribut elements

Name	Type	Cardinality
Travée	Numeric	1-1
Rayon	Numeric	1-1
Etage	Numeric	1-1

2.4.4.3.1. Travée

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Travée		Numeric	1-1		No	Yes	Comp att.:Localisation

2.4.4.3.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Loc_Travée

2.4.4.3.2. Rayon

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Rayon		Numeric	1-1		No	Yes	Comp att.:Localisation

2.4.4.3.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Loc_Rayon

2.4.4.3.3. Etage

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Etage		Numeric	1-1		No	Yes	Comp att.:Localisation

2.4.4.3.3.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Loc_Etage

2.4.5. RELATIONSHIP TYPES

- Relationship types list

Name
de
emprunte

2.4.5.1. de

Entity in relation
OUVRAGE

2.4.5.2. emprunte

Entity in relation
EMPRUNTEUR
PROJET

2.4.6. GROUPS

2.4.6.1. Group: 1

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
Numéro-série	0-1	Primary	No	No	No	No

2.4.6.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	1

2.4.7. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	id26

2.5. OUVRAGE

2.5.1. GENERAL INFORMATION

- Name: OUVRAGE
- Short name: BOOK
- Technical Identification: id7
- In Cluster: DOCUMENT.DOCUMENT

  Type: Disjoint

  Super type is : DOCUMENT

2.5.2. DESCRIPTIONS

Type	Description
Semantic	A book is any written piece of work in the literature, science or technical domain.

2.5.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	OUVRAGE

2.5.4. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
ISBN	Numeric	1-1
Editeur	Char	1-1

2.5.4.1. ISBN

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
ISBN		Numeric	1-1	set	No	Yes	Entity: OUVRAGE

2.5.4.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	ISBN

2.5.4.2. Editeur

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Editeur		Char	1-1		No	Yes	Entity: OUVRAGE

2.5.4.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Editeur

2.5.5. RELATIONSHIP TYPES

- Relationship types list

Name
de

2.5.5.1. de

Entity in relation
EXEMPLAIRE

2.5.6. GROUPS

2.5.6.1. Group: 1

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
ISBN	0-1	Secondary	No	No	No	No

2.5.6.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	1

2.5.7. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	id7

2.6. PROJET

2.6.1. GENERAL INFORMATION

- Name: PROJET
- Short name: PRO
- Technical Identification: id36

2.6.2. DESCRIPTIONS

Type	Description
Semantic	A project is an activity that has been given proper financial resources.

2.6.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	PROJET

2.6.4. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
CodeProjet	Char	1-1
Nom	Char	1-1
Société	Char	1-1

2.6.4.1. CodeProjet

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
CodeProjet		Char	1-1		No	Yes	Entity: PROJET
Description: Short name or acronym commonly used to designate the project.							

2.6.4.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	CodeProjet
Simple Attribut	CodeProjet
Simple Attribut	CodeProjet

2.6.4.2. Nom

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Nom		Char	1-1		No	Yes	Entity: PROJET
Description: Complete name of the project.							

2.6.4.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Nom

2.6.4.3. Société

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Société		Char	1-1		No	Yes	Entity: PROJET

2.6.4.3.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Société

2.6.5. RELATIONSHIP TYPES

- Relationship types list

Name
occupé
emprunte

2.6.5.1. occupé

Entity in relation
EMPRUNTEUR

2.6.5.2. emprunte

Entity in relation
EXEMPLAIRE
EMPRUNTEUR

2.6.6. GROUPS

2.6.6.1. Group: 1

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
CodeProjet	0-1	Primary	No	No	No	No

2.6.6.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	1

2.6.6.2. Group: 2

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
Nom	0-1	Secondary	No	No	No	No

2.6.6.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	2

2.6.7. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	id36

2.7. RAPPORT

2.7.1. GENERAL INFORMATION

- Name: RAPPORT
- Technical Identification: id892
- In Cluster: DOCUMENT.DOCUMENT

  Type: Disjoint

  Super type is : DOCUMENT

2.7.2. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	RAPPORT

2.7.3. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
Code-Rapport	Char	1-1
Projet	Char	1-1

2.7.3.1. Code-Rapport

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Code-Rapport		Char	1-1		No	Yes	Entity: RAPPORT

2.7.3.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Code-Rapport

2.7.3.2. Projet

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Projet		Char	1-1		No	Yes	Entity: RAPPORT

2.7.3.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Projet

2.7.4. GROUPS

2.7.4.1. Group: IDREPORT

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
Code-Rapport	0-1	Secondary	No	No	No	No

2.7.4.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	IDREPORT

2.7.5. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	id892

3. Relationship Types

3.1. de

3.1.1. GENERAL INFORMATION

- Name: de
- Short name: OF
- Technical Identification: id64

3.1.2. DESCRIPTIONS

Type	Description
Semantic	Specifies the book of each copy.

3.1.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	FKde

3.1.4. ROLES

Card	Name
0-N	OUVRAGE
1-1	EXEMPLAIRE

3.1.5. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	64

3.2. emprunte

3.2.1. GENERAL INFORMATION

- Name: emprunte
- Short name: CLO
- Technical Identification: id54

3.2.2. DESCRIPTIONS

Type	Description
Semantic	History of a former borrowing. When a copy is brought back, the information on its borrowing is kept, making up a so-called "closed borrowing". There cannot exist more than one closed borrowing for a given copy at the same date.

3.2.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	emprunte

3.2.4. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
Date-Emprunt	Date	1-1
Date-Retour	Date	0-1

3.2.4.1. Date-Emprunt

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Date-Emprunt		Date	1-1		No	Yes	Relation:emprunte
Description: Date on which the copy were borrowed.							

3.2.4.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Date-Emprunt

3.2.4.2. Date-Retour

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
Date-Retour		Date	0-1		No	Yes	Relation:emprunte
Description: Date on which the copy was brought back.							

3.2.4.2.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	Date-Retour

3.2.5. ROLES

Card	Name
0-N	EXEMPLAIRE
0-N	EMPRUNTEUR
0-N	PROJET

3.2.6. GROUPS

3.2.6.1. Group: 1

Attribut Composition	Card	Identifier	Access Key	Coexistence	At-least-1	Exclusive
Date-Emprunt	0-1	Primary	No	No	No	No

3.2.6.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	1

3.2.7. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	54

3.3. occupé

3.3.1. GENERAL INFORMATION

- Name: occupé
- Technical Identification: id915

3.3.2. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	FKoccupé

3.3.3. ROLES

Card	Name
0-1	EMPRUNTEUR
0-N	PROJET

3.3.4. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	915

3.4. responsable

3.4.1. GENERAL INFORMATION

- Name: responsable
- Short name: RESP
- Technical Identification: id69

3.4.2. DESCRIPTIONS

Type	Description
Semantic	Specifies for each borrower a possible other borrower who can be contacted by the library whenever the former cannot be joined.

3.4.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Group	FKresponsable

3.4.4. ROLES

Card	Name
0-1	EMPRUNTEUR
0-N	EMPRUNTEUR

3.4.5. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	69

3.5. réserve

3.5.1. GENERAL INFORMATION

- Name: réserve
- Technical Identification: id909

3.5.2. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	réserve

3.5.3. ATTRIBUTS

- Attribut list

Name	Type	Cardinality
DateRéservation	Char	1-1

3.5.3.1. DateRéservation

Name	Short Name	Type	Card	Set Type	Stable	Recycl	Owner
DateRéservation		Char	1-1		No	Yes	Relation:réserve

3.5.3.1.1. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Simple Attribut	DateRéservation

3.5.4. ROLES

Card	Name
0-N	EMPRUNTEUR
0-N	DOCUMENT

3.5.5. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	909

3.6. écrit

3.6.1. GENERAL INFORMATION

- Name: écrit
- Short name: WRIT
- Technical Identification: id74

3.6.2. DESCRIPTIONS

Type	Description
Semantic	Associates each author with the books s/he has written.

3.6.3. TRACE FORWARD BACKWARD

No backward trace

Forward

Type	Name
Entity-type	écrit

3.6.4. ROLES

Card	Name
1-N	AUTEUR
0-N	DOCUMENT

3.6.5. META PROPERTIES

Name	Value
Parent_OID	0
Copy_OID	74

Chapter 4. DOCUMENT: biblio.txt

1. General information

- Name: biblio.txt
- Version: Texte1
- Type of file: texte
- Path: C:\Docbook\monDocbook\Biblio\
- Creation date: 31-12-2003
- Modification Date: 31-12-2003

2. Texte

1 : Rapport d'interview
2 :
3 : Les faits rapportés ci-dessous ont été renseignés par les employés gérant
4 : la salle des livres de la bibliothèque. Leurs fonctions sont de conseiller
5 : et d'aider les usagers, de veiller au rangement des livres, de gérer les emprunts.
6 :
7 : <Le contenu de la bibliothèque
8 : Un ouvrage est une oeuvre littéraire, scientifique ou technique publiée.
9 : Il est caractérisé par son numéro identifiant, son titre, son éditeur, sa
10 : date de première parution, ses mots-clés (10 au maximum), une brève note
11 : de présentation (ces notes sont en cours de constitution), le nom de ses
12 : auteurs et ses références bibliographiques (c'est-à-dire les autres ouvrages
13 : qu'il cite). A un ouvrage correspondent un certain nombre d'exemplaires, qui
14 : en sont la matérialisation physique. La bibliothèque acquiert des exemplaires
15 : d'ouvrages et les met en consultation et en prêt. Les exemplaires d'un
16 : ouvrage possèdent un numéro d'ordre qui les distingue parmi tous les
17 : exemplaires de cet ouvrage. Chaque exemplaire est en outre caractérisé par
18 : sa date d'acquisition, sa localisation physique (c'est-à-dire l'étage, la
19 : travée et le rayon), son emprunteur éventuel, le nombre de livres qui le
20 : constituent (on doit les emprunter tous ensembles), son état de vétusté
21 : (codé par une lettre) et par un commentaire éventuel sur cet état. On
22 : signale que l'auteur d'un ouvrage est caractérisé par ses nom, prénom,
23 : date de naissance et origine. Il se peut que seule la première information
24 : soit connue pour certains auteurs. On ne retient que les auteurs d'ouvrages
25 : enregistrés dans la bibliothèque.
26 :
27 : Les emprunts
28 : Un exemplaire peut être emprunté, depuis une date déterminée, par un
29 : emprunteur. Ce dernier est identifié par un numéro matricule. On en
30 : connaît le nom, le prénom, l'adresse (nom de la société, voie, code postal
31 : et nom de la ville) et les numéros de téléphone auxquels il est possible de
32 : le joindre, ainsi que, si possible, les coordonnées d'un autre emprunteur qui
33 : peut répondre pour le premier en cas d'absence. Un exemplaire restitué est
34 : remis en rayon à la fin de la journée et ne pourra être à nouveau emprunté que
35 : le jour ouvrable suivant. Quand un emprunteur emprunte un exemplaire d'un ouvrage,
36 : il le fait pour le compte d'un projet (décrit par un code et un nom de projet,
37 : tous deux identifiants individuellement). Cette information est importante pour
38 : l'imputation des frais des emprunts. Quand un exemplaire est restitué, on conserve
39 : les informations sur cet emprunt en y ajoutant la date de restitution.ÿ

Index